Annexe A

Corpus et jeux d’essais

A.1 Structures descriptives

Nous donnons ci-dessous les structures descriptives des éléments du lexique que nous avons utilisé pour les jeux d’essais de PELEAS, c’est-à-dire *bouche*, *fille*, *fils*, *frère*, *jeune*, *langue*, *maison*, *manger*, *mère*, *mort*, *nez*, *peau*, *père*, *roi*, *sœur*, *soleil*, *système*, *toucher*, *vie*, *vieux* et *voir*.

bouche

```
%%PDL
ENTRY(bouche) = BEGIN
  quotidien = BEGIN
    anatomie = BEGIN
      cavité_buccale = BEGIN
        contenu;
        appétit;
        parole;
        parole =-=> contenu;
      END;
      lèvres;
    END;
    personne = BEGIN
      convive;
      personne_à_charge;
      gourmet;
      personne_difficile;
      personne_à_charge ===> convive;
      gourmet ===> personne_difficile;
    END;
    moment =
      fin;
  END;
  militaire = BEGIN
    arme =
      canon = BEGIN
        explosion;
        bruit;
        chaleur;
      END;
    composant =
      orifice = BEGIN
        arme_à_feu;
        projectile;
        détonation;
```

```
      END;
  END;
  géographique =
    extrémité = BEGIN
      embouchure = BEGIN
        fleuve;
        mer;
      END;
      entrée = BEGIN
        golfe;
        détroit;
        mer;
      END;
      embouchure =-=> entrée;
      entrée =-=> embouchure;
    END;
  technique = BEGIN
    trou = BEGIN
      ouverture = BEGIN
        four;
        enceinte;
        profondeur;
        four ===> enceinte;
      END;
      accès = BEGIN
        cave;
        métro;
        cave =-=> métro;
        métro =-=> cave;
      END;
    END;
    ustensile =
      bouche_d_incendie = BEGIN
        pompiers;
        eau;
        pression;
        incendie;
        incendie ===> pompiers;
        pompiers ===> eau;
        eau ===> pression;
      END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  BELLE_BOUCHE : PrecedeParAdjectifDEvaluationEsthetique => {
    (quotidien.anatomie.lèvres Salient)
  }
  FINE_BOUCHE : ComplementDeVerbeDEtat AND PrecedeParFine => {
    (quotidien.personne.gourmet Salient)
  }
  FAIRE_LA_FINE_BOUCHE : (PrecedeParFine AND ComplementDeFaire) AND
  PrecedeParArticleDefini => {
    (quotidien.personne.personne_difficile Salient)
  }
  FERMER_OUVRIR : ComplementDeOuvrirFermer AND PrecedeParArticleDefini => {
    (quotidien.anatomie.cavité_buccale.parole Salient)
    (quotidien.anatomie.lèvres Salient)
  }
  EAU_A_LA_BOUCHE : PrecedeParEau AND PrecedeParArticleDefini => {
    (quotidien.anatomie.cavité_buccale.appétit Salient)
  }
  DANS_LA_BOUCHE : PrecedeParPlein OR (PrecedeParDans AND PrecedeParArticleDefini) => {
    (quotidien.anatomie.cavité_buccale.contenu Salient)
  }
  A_TABLE : SuiviParATable => {
    (quotidien.personne.convive Salient)
```

```
}
A_NOURRIR : SuiviParNourrir => {
  (quotidien.personne.personne_à_charge Salient)
}
BONNE_BOUCHE : PrecedeParBonne => {
  (quotidien.moment.fin Salient)
}
BOUCHE_A_FEU : SuiviParAFeu => {
  (militaire.arme.canon Salient)
}
ARME1 : VocabulaireMilitaire => {
  (militaire.composant.orifice Salient)
}
DE_FLEUVE : CompleteParNomDeFleuveGenitif => {
  (géographique.extrémité.embouchure Salient)
}
DE_GOLFE : CompleteParNomDeLieuCotierGenitif => {
  (géographique.extrémité.entrée Salient)
}
APPETIT : SynonymeDAppetitDansLeContexte => {
  (quotidien.anatomie.cavité_buccale.appétit Salient)
}
PAROLE : SynonymeDeMotDansLeContexte => {
  (quotidien.anatomie.cavité_buccale.parole Salient)
}
EXPLOSION_BRUIT : SynonymeDeExplosionBruitDansLeContexte => {
  (militaire.arme.canon.explosion Salient)
  (militaire.arme.canon.bruit Salient)
}
ARME2 : SynonymeDeArmeAFeuProjectileDetonationDansLeContexte => {
  (militaire.composant.orifice.arme_à_feu Salient)
  (militaire.composant.orifice.projectile Salient)
  (militaire.composant.orifice.détonation Salient)
}
FLEUVE : FleuveDansLeContexte => {
  (géographique.extrémité.embouchure.fleuve Salient)
}
GOLFE : GolfeDansLeContexte => {
  (géographique.extrémité.entrée.golfe Salient)
}
DETROIT : DetroitDansLeContexte => {
  (géographique.extrémité.entrée.détroit Salient)
}
FOUR : FourDansLeContexte => {
  (technique.trou.ouverture.four Salient)
}
ENCEINTE : SynonymeDeEnceinteDansLeContexte => {
  (technique.trou.ouverture.enceinte Salient)
}
CAVE : SynonymeDeCaveDansLeContexte => {
  (technique.trou.accès.cave Salient)
}
METRO : SynonymeDeMetroDansLeContexte => {
  (technique.trou.accès.métro Salient)
}
POMPIER : PompiersDansLeContexte => {
  (technique.ustensile.bouche_d_incendie.pompiers Salient)
}
INCENDIE : SynonymeDeIncendieDansLeContexte => {
  (technique.ustensile.bouche_d_incendie.incendie Salient)
}
DEFAUT : PositionNom => {
  (quotidien.anatomie.cavité_buccale Salient)
}
INANIME : PositionNom AND CompleteParUnInanimeAuGenitif => {
  (technique.trou.ouverture.profondeur Salient)
```

```
  }
END.
```

fille

```
%%PDL
ENTRY(fille) = BEGIN
  quotidien = BEGIN
    personne = BEGIN
      femme = BEGIN
        enfant;
        jeune;
      END;
      enfant;
    END;
    manière =
      sans_prévenir =
        partir;
    statut = BEGIN
      mariée;
      mère;
      servante;
      religieuse;
      prostituée;
    END;
    hiérarchie = BEGIN
      successeur;
      généré;
      similaire;
      généré ===> successeur;
      généré ===> similaire;
      successeur ===> similaire;
    END;
    attitude = BEGIN
      bienveillance;
      ingratitude;
      obéissance;
      insulte;
    END;
    titre =
      insulte;
  END;
  littéraire =
    descendance = BEGIN
      descendant;
      originaire_de;
    END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  FILS_FILLE : FilsDansLeContexte => {
    (quotidien.personne.enfant Salient)
  }
  FILS_DES : PremierDansGroupeNdeN AND (AutreMembreNdeNEstNomPropre AND
  AutreMembreNdeNPluriel) => {
    (littéraire.descendance.descendant Salient)
  }
  VIENT_DE : PremierDansGroupeNdeN AND (NOT(AutreMembreNdeNPluriel) AND
  AutreMembreNdeNEstNomPropre) => {
    (littéraire.descendance.descendant Salient)
    (littéraire.descendance.originaire_de Salient)
  }
  INSULTE : PremierDansGroupeNdeN AND AutreMembreNdeNEstUneInsulte => {
```

```
    (quotidien.personne.femme Salient)
    (quotidien.titre.insulte Salient)
  }
  BONNE_FILLE : AdjectifDEvaluationPositive => {
    (quotidien.attitude.bienveillance Salient)
  }
  MAUVAISE_FILLE : AdjectifDEvaluationNegative => {
    (quotidien.attitude.ingratitude Salient)
  }
  PETITE_FILLE : QualifieParUnAdjectifDeTaille => {
    (quotidien.personne.femme.enfant Salient)
  }
  JEUNE_FILLE : QualifieParJeune => {
    (quotidien.personne.femme.jeune Salient)
  }
  VIEILLE_FILLE : QualifieParVieille OR ObjetDeRester => {
    (quotidien.statut.mariée Negated)
  }
  FILLE_MERE : QualifieParMere => {
    (quotidien.statut.mariée Negated)
    (quotidien.statut.mère Negated)
  }
  FILLE_DE_SALLE : PremierDansGroupeNdeN AND AutreMembreNdeNEstNomDeLieu => {
    (quotidien.statut.servante Salient)
  }
  FILLE_DU_CALVAIRE : PremierDansGroupeNdeN AND AutreMembreNdeNEstCongregationReligieuse => {
    (quotidien.statut.religieuse Salient)
  }
  PROSTITUEE : (PremierDansGroupeNdeN AND AutreMembreNdeNEstJoie) OR QualifieParPublique => {
    (quotidien.statut.prostituée Salient)
  }
  ADJECTIF : PositionAdjectif => {
    (quotidien.hiérarchie.généré Salient)
    (quotidien.hiérarchie.successeur Salient)
  }
  FILLE_DE_LAIR : SuiviParDeLAir => {
    (quotidien.personne.femme Valid)
    (quotidien.personne.enfant Valid)
    (quotidien.manière.sans_prévenir.partir Salient)
  }
  NOM : PositionNom => {
    (quotidien.personne.femme Salient)
    (quotidien.personne.enfant Salient)
  }
END.
```

fils

```
%%PDL
ENTRY(fils) = BEGIN
  quotidien = BEGIN
    personne =
      garçon;
    hiérarchie = BEGIN
      successeur;
      généré;
      similaire;
      généré ===> similaire;
      généré ===> successeur;
    END;
    attitude = BEGIN
      ingratitude;
      bienveillance;
```

```
      obéissance;
      ingratitude <=/=> bienveillance;
      bienveillance ===> obéissance;
    END;
    titre =
      insulte;
  END;
  littéraire =
    descendance = BEGIN
      descendant;
      originaire_de;
    END;
  religion =
    personne = BEGIN
      dieu;
      christ =
        homme;
    END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  FILS_A_PAPA : SuiviParAPapa => {
    (quotidien.attitude.ingratitude Salient)
    (quotidien.hiérarchie.successeur Salient)
  }
  CHRIST : SaintEspritDansLeContexte OR (PereMajusculeDansLeContexte OR
  (InitialeMajuscule AND (PrecedeParArticleDefini OR CompleteParDeLHomme)))
  => {
    (religion.personne.christ Salient)
  }
  FILS_DES : PremierDansGroupeNdeN AND (AutreMembreNdeNEstNomPropre AND
  AutreMembreNdeNPluriel) => {
    (littéraire.descendance.descendant Salient)
  }
  VIENT_DE : PremierDansGroupeNdeN AND (NOT(AutreMembreNdeNPluriel) AND
  AutreMembreNdeNEstNomPropre) => {
    (littéraire.descendance.descendant Salient)
    (littéraire.descendance.originaire_de Salient)
  }
  INSULTE : PremierDansGroupeNdeN AND AutreMembreNdeNEstUneInsulte => {
    (quotidien.personne.garçon Salient)
    (quotidien.titre.insulte Salient)
  }
  NOM : PositionNom => {
    (quotidien.personne.garçon Salient)
  }
  ADJECTIF : PositionAdjectif => {
    (quotidien.hiérarchie.successeur Salient)
    (quotidien.hiérarchie.généré Salient)
  }
  PERE_FILS : PereDansLeContexte => {
    (quotidien.hiérarchie.similaire Salient)
  }
  BON_FILS : AdjectifDEvaluationPositive => {
    (quotidien.attitude.bienveillance Salient)
  }
  MAUVAIS_FILS : AdjectifDEvaluationNegative => {
    (quotidien.attitude.ingratitude Salient)
  }
END.
```

frère

```
%%PDL
ENTRY(frère) = BEGIN
  quotidien = BEGIN
    personne =
      homme = BEGIN
        masculin;
        famille;
        fratrie;
        fratrie ===> famille;
      END;
    sentiment = BEGIN
      ami;
      proche;
      compagnon;
      hypocryte;
      compagnon ===> proche;
      ami ===> proche;
      hypocryte <=/=> ami;
    END;
    hiérarchie = BEGIN
      simultané;
      similaire;
      solidaire;
      simultané ===> similaire;
    END;
  END;
  religion =
    personne =
      religieux;
  littéraire =
    titre =
      membre = BEGIN
        ordre;
        association;
      END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  VOCATIF : CasVocatif => {
    (religion.personne.religieux Salient)
    (littéraire.titre.membre Salient)
  }
  MON_FRERE : PositionNom AND PrecAdjPoss => {
    (quotidien.personne.homme.fratrie Salient)
    (quotidien.sentiment.ami Salient)
  }
  FRERE_DE_X : PositionNom AND CompleteParNPAuGenitif => {
    (quotidien.personne.homme.fratrie Salient)
  }
  FAUX_FRERE : QualifieParFaux => {
    (quotidien.sentiment.hypocryte Salient)
  }
  FRERE_DE_SANG : PositionNom AND (AutreMembreSNdeSNEstUnInanime AND
  PremierDansGroupeSNdeSN) => {
    (quotidien.sentiment.compagnon Salient)
    (quotidien.hiérarchie.solidaire Salient)
  }
  RELIGIEUX : ThemeMythiqueOuReligieux OR (PositionNom AND SuiviParNomPropre)
  => {
    (religion.personne.religieux Salient)
  }
  ADJECTIF : PositionAdjectif => {
    (quotidien.hiérarchie.simultané Salient)
    (quotidien.hiérarchie.solidaire Salient)
```

```
  }
  DEFAUT : _TRUE => {
    (quotidien.personne.homme.fratrie Salient)
    (quotidien.sentiment.ami Salient)
  }
END.
```

jeune

```
%%PDL
ENTRY(jeune) = BEGIN
  technique =
    état = BEGIN
      immature;
      fini;
      insuffisant;
      récent;
      immature <=/=> fini;
      insuffisant =-=> fini;
    END;
  quotidien = BEGIN
    personne = BEGIN
      individu;
      collectif;
      collectif =-=> individu;
      individu =-=> collectif;
    END;
    animal =
      individu;
    état = BEGIN
      âgé;
      immature;
      âgé <=/=> immature;
    END;
    attitude = BEGIN
      vigoureux;
      charmant;
      naïf;
    END;
    relation =
      cadet;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  JEUNE_H : QualifieUnePersonne => {
    (quotidien.personne.individu Salient)
    (quotidien.état.âgé Negated)
  }
  JEUNE_A : QualifieUnAnimal => {
    (quotidien.animal.individu Salient)
    (quotidien.état.âgé Negated)
  }
  FONCTION : QualifieUneFonctionOuProfession => {
    (quotidien.état.âgé Negated)
  }
  ACTIVITE : QualifieUneActivite => {
    (technique.état.fini Negated)
    (technique.état.insuffisant Salient)
  }
  PRODUIT : QualifieUnProduitDActivite => {
    (technique.état.insuffisant Salient)
  }
```

```
  INANIME : QualifieUnInanime => {
    (technique.état.fini Negated)
    (technique.état.immature Salient)
    (technique.état.récent Salient)
  }
  NP : QualifieUnNomPropre => {
    (quotidien.relation.cadet Salient)
  }
  AGE_JEUNE : IncoherenceSemantiqueAgeJeune => {
    (quotidien.attitude.vigoureux Salient)
    (quotidien.attitude.charmant Salient)
  }
  ADJECTIF : PositionAdjectif => {
    (quotidien.état.âgé Negated)
  }
  SING : NOT(Pluriel) AND PositionNom => {
    (quotidien.personne.individu Salient)
    (quotidien.état.âgé Negated)
  }
  PLUR : Pluriel AND PositionNom => {
    (quotidien.personne.collectif Salient)
    (quotidien.état.âgé Negated)
  }
END.
```

langue

```
%%PDL
ENTRY(langue) = BEGIN
  linguistique =
    système = BEGIN
      parole = BEGIN
        parlée;
        argotique;
      END;
      symboles;
    END;
  anatomie =
    organe =
      organe = BEGIN
        déglutition;
        parole;
        goût;
      END;
  quotidien = BEGIN
    objet =
      partie =
        allongée;
    état =
      dans_le_besoin;
    activité =
      se_moquer;
    attitude = BEGIN
      bavard;
      discret;
      silencieux;
      médisant;
      rigide;
    END;
  END;
  littéraire =
    activité =
      pourparlers;
```

```
  cuisine =
    plat =
      plat;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  SE_MOQUER : (VerbeIntransitif AND PrecedeParArticleDefini) AND ObjetDeTirer => {
    (quotidien.activité.se_moquer Salient)
  }
  EN_BAVER : (VerbeTransitif AND PrecedeParArticleDefini) AND ObjetDeTirer => {
    (quotidien.état.dans_le_besoin Salient)
  }
  BAVARD : QualifieParBienPendue => {
    (quotidien.attitude.bavard Salient)
  }
  DISCRET : PrecAdjPoss AND ObjetDeTenir => {
    (quotidien.attitude.discret Salient)
  }
  LONGUE : QualifieParLongue => {
    (quotidien.attitude.discret Negated)
  }
  SE_TAIRE : PrecAdjPoss AND ObjetDeAvaler => {
    (quotidien.attitude.silencieux Salient)
  }
  POURPARLERS : NOT(PrecedeParArticleDefini) AND ObjetDePrendre => {
    (littéraire.activité.pourparlers Salient)
  }
  MAUVAISE : AdjectifDEvaluationNegative => {
    (quotidien.attitude.médisant Salient)
  }
  BOIS : AutreMembreNdeNEstBois AND PremierDansGroupeNdeN => {
    (linguistique.système.parole Salient)
    (quotidien.attitude.rigide Salient)
  }
  INANIME : PremierDansGroupeSNdeSN AND AutreMembreSNdeSNEstUnInanime => {
    (quotidien.objet.partie.allongée Salient)
  }
  VIVANTE : QualifieParVivante => {
    (linguistique.système.parole.parlée Salient)
  }
  MORTE : QualifieParMorte => {
    (linguistique.système.parole.parlée Negated)
  }
  ARGOT : QualifieParVerte => {
    (linguistique.système.parole.argotique Salient)
  }
  FORMELLE : QualifieParFormelle => {
    (linguistique.système.symboles Salient)
    (quotidien.attitude.rigide Salient)
  }
  PAROLE : ObjetDeParler OR (ObjetDeConnaitre OR
  (VocabulaireLinguistiqueOuParole OR (QualifieParUnOrdinal OR
  QualifieParMaternelle))) => {
    (linguistique.système.parole Salient)
  }
  NOM : PositionNom => {
    (anatomie.organe.organe Salient)
  }
END.
```

maison

```
%%PDL
```

```
ENTRY(maison) = BEGIN
  quotidien = BEGIN
    édifice = BEGIN
      bâtiment;
      habitation;
      édifice_spécialisé;
      habitation ===> bâtiment;
      édifice_spécialisé ===> bâtiment;
      habitation <=/=> édifice_spécialisé;
    END;
    logement =
      foyer;
  END;
  astrologie =
    zone =
      division_céleste;
  économie = BEGIN
    artisanat = BEGIN
      entreprise;
      artisanal;
      artisanal ===> entreprise;
    END;
    statut = BEGIN
      interne;
      famille =
        domestique;
      interne <=/=> famille;
    END;
  END;
  littéraire =
    groupe_social = BEGIN
      famille = BEGIN
        membre;
        honorable;
        noblesse;
      END;
      serviteur = BEGIN
        souverain;
        civil;
        militaire;
        collectif;
        civil <=/=> militaire;
      END;
      famille <=/=> serviteur;
    END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  NOBLESSE : PrecedeParArticleDefini AND (PremierDansGroupeNdeN AND
  AutreMembreNdeNEstNomPropre) => {
    (littéraire.groupe_social.famille.noblesse Salient)
  }
  EDIFICE_SPECIALISE : PremierDansGroupeNdeN AND
  (AutreMembreNdeNEstInstitution OR AutreMembreNdeNEstJeunes) => {
    (quotidien.édifice.édifice_spécialisé Salient)
  }
  ENTREPRISE : ThemeEconomique OR (SuiviParNomPropre OR
  (PremierDansGroupeNdeN AND (AutreMembreNdeNEstSubstantif OR
  AutreMembreNdeNEstProduitCommercial))) => {
    (économie.artisanat.entreprise Salient)
  }
  SOUVERAIN : PremierDansGroupeNdeN AND AutreMembreNdeNEstSynonymeDeSouverain => {
    (littéraire.groupe_social.serviteur Salient)
  }
  DOMESTIQUE : SecondDansGroupeNdeN AND AutreMembreNdeNEstGens => {
    (économie.statut.famille.domestique Salient)
  }
```

```
  CIVILE : QualifieParCivileEnApposition => {
    (littéraire.groupe_social.serviteur.civil Salient)
  }
  MILITAIRE : QualifieParMilitaireEnApposition => {
    (littéraire.groupe_social.serviteur.militaire Salient)
  }
  HONNEUR1: AdjectifDEvaluationPositive => {
    (littéraire.groupe_social.famille.honorable Salient)
    (littéraire.groupe_social.famille.noblesse Salient)
    (économie.artisanat.entreprise Salient)
  }
  HONNEUR2: AdjectifDEvaluationNegative OR (AdjectifDEvaluationPositive
  AND PrecedeParNegation) => {
    (littéraire.groupe_social.famille.honorable Negated)
    (économie.artisanat.entreprise Negated)
  }
  MEMBRE : (SecondDansGroupeNdeN AND (PrecedeParArticleDefini AND
  AutreMembreNdeNEstUnRoleFamilial)) OR (VerbeDEtatDansLeContexte OR
  ObjetDeFairePartie) => {
    (littéraire.groupe_social.famille.membre Salient)
  }
  FOYER : PrecedeParALa => {
    (quotidien.logement.foyer Salient)
  }
  TP : VocabulaireTravauxPublics => {
    (quotidien.édifice.habitation Salient)
  }
  ARTISANAT : EnApposition AND QualifieProduitCommercial => {
    (économie.artisanat.artisanal Salient)
  }
  INTERNE : EnApposition AND QualifieUnAnime => {
    (économie.statut.interne Salient)
  }
  BOUYGUES : QualifieParIndividuelle => {
    (quotidien.édifice.habitation Salient)
  }
  ATTRIB : SupposeEtreAnime OR (QualifieParMere OR (PrecedeParArticleDefini
  AND (PrecedeParPour OR PrecedeParSur))) => {
    (économie.artisanat.entreprise Salient)
    (littéraire.groupe_social.famille Salient)
  }
  DEFAUT : _TRUE => {
    (quotidien.édifice.habitation Valid)
    (quotidien.logement.foyer Valid)
  }
END.
```

manger

```
%%PDL
ENTRY(manger) = BEGIN
  quotidien = BEGIN
    absorption = BEGIN
      se_nourrir =
        aliment;
      nourriture;
      repas;
      appétit;
      nourriture ===> se_nourrir;
      repas ===> nourriture;
      appétit ===> se_nourrir;
    END;
    destruction = BEGIN
```

```
      révéler = BEGIN
        secret;
        aveux;
        délation;
      END;
      dépenser = BEGIN
        vainement;
        perte;
      END;
      révéler <=/=> dépenser;
    END;
    manière = BEGIN
      destructif;
      aggressif;
      destructif ===> aggressif;
    END;
    occultation = BEGIN
      oublier;
      prononcer;
      cacher;
      oublier =-=> prononcer;
    END;
    émotion = BEGIN
      désir;
      trouble;
      désir ===> trouble;
    END;
  END;
  technique = BEGIN
    absorption =
      consommer = BEGIN
        carburant;
        fonctionnement;
      END;
    destruction = BEGIN
      entamer;
      ronger;
      abîmer;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  MIAM : FonctionVerbe AND ObjetEstUnAliment => {
    (quotidien.absorption.se_nourrir Salient)
    (quotidien.manière.destructif Salient)
  }
  VOITURE : FonctionVerbe AND SujetEstUnInanimeConcret => {
    (quotidien.manière.destructif Salient)
    (technique.absorption.consommer.carburant Salient)
  }
  UN_MORCEAU : FonctionVerbe AND ObjetEstUnSynonymeDePetitOuPeu => {
    (quotidien.absorption.se_nourrir Salient)
    (quotidien.manière.destructif Salient)
  }
  INFO : FonctionVerbe AND ObjetEstUneInformationOuUnMedia => {
    (quotidien.occultation.oublier Salient)
  }
  MOT : FonctionVerbe AND ObjetContientMot => {
    (quotidien.occultation.prononcer Negated)
  }
  LE_MORCEAU : FonctionVerbe AND ObjetEstLeMorceau => {
    (quotidien.destruction.révéler Salient)
  }
  DILAPIDER : FonctionVerbe AND (ObjetEstCardinal OR ObjetEstSynonymeDeSommeDArgent) => {
    (quotidien.destruction.dépenser Salient)
  }
```

```
  DES_YEUX : FonctionVerbe AND AdverbeEstDesYeux => {
    (quotidien.émotion.désir Salient)
  }
  TROUBLE : FonctionVerbe AND (SujetEstUnInanimeAbstrait AND ObjetEstUnAnimeOuUnePartie) => {
    (quotidien.manière.aggressif Salient)
    (quotidien.émotion.trouble Salient)
  }
  RONGER : FonctionVerbe AND (ObjetEstUnInanimeConcret AND SujetEstUnInanimeConcret) => {
    (technique.destruction.ronger Salient)
  }
  LE_MANGER : PositionNom => {
    (quotidien.absorption.nourriture Salient)
    (quotidien.absorption.repas Salient)
    (quotidien.absorption.appétit Salient)
  }
  PAS_CHER : DansExpressionCaNeMangePasDePain => {
    (quotidien.destruction.dépenser Negated)
  }
  DEFAUT : _TRUE => {
    (quotidien.absorption.se_nourrir Salient)
    (quotidien.manière.destructif Salient)
    (technique.absorption.consommer Salient)
  }
  CACHER : FonctionVerbe AND (ObjetEstVisage OR ObjetEstUnAstre) => {
    (quotidien.occultation.cacher Salient)
  }
END.
```

mère

```
%%PDL
ENTRY(mère) = BEGIN
  quotidien = BEGIN
    titre =
      madame = BEGIN
        péjoratif;
        âgé;
      END;
    personne = BEGIN
      femme = BEGIN
        enfants;
        famille;
      END;
      partenaire_sexuel =
        fertile;
      épouse;
    END;
    hiérarchie = BEGIN
      générateur;
      génitrice;
      antécédant;
      similaire;
      principale;
      génitrice ===> générateur;
      générateur ===> antécédant;
      générateur ===> similaire;
      générateur ===> principale;
    END;
    attitude = BEGIN
      humanité;
      bienveillance;
      protection;
      bienveillance =+=> protection;
```

```
    END;
  END;
  littéraire =
    hiérarchie = BEGIN
      cause;
      origine;
    END;
  religion = BEGIN
    transcendance =
      divinité;
    personne =
      religieuse;
  END;
  cuisine =
    instrument =
      mère_du_vinaigre;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  DE_FAMILLE : SuiviParDeFamille => {
    (quotidien.personne.femme.famille Salient)
    (quotidien.hiérarchie.génitrice Salient)
    (quotidien.personne.épouse Salient)
  }
  LA_MERE_X : PrecedeParArticleDefini AND SuiviParNomPropre => {
    (quotidien.titre.madame.péjoratif Salient)
  }
  MERE_THERESA : NOT(PrecedeParArticleDefini) AND SuiviParNomPropre => {
    (religion.personne.religieuse Salient)
  }
  MERE_SUP : PositionNom AND QualifieParSuperieure => {
    (religion.personne.religieuse Salient)
    (quotidien.hiérarchie.principale Salient)
  }
  CUISINE : VinaigreDansLeContexte OR (VocabulaireCulinaire AND
  EstSupposeInanime) => {
    (quotidien.hiérarchie.générateur Salient)
    (cuisine.instrument.mère_du_vinaigre Salient)
  }
  MAISON_MERE : PositionAdjectif AND QualifieUnInanime => {
    (quotidien.hiérarchie.générateur Salient)
    (quotidien.hiérarchie.principale Salient)
  }
  FEMELLE : PositionAdjectif AND QualifieUnAnime => {
    (quotidien.hiérarchie.génitrice Salient)
  }
  MERE_PATRIE : PositionAdjectif AND EstAttributDUnLieu => {
    (littéraire.hiérarchie.origine Salient)
    (quotidien.hiérarchie.principale Salient)
  }
  LIEU : ((PositionNom AND EstAttributDUnLieu) AND PremierDansGroupeNdeN)
  AND AutreMembreNdeNEstUnInanime => {
    (littéraire.hiérarchie.origine Salient)
  }
  CAUSE : ((PositionNom AND NOT(EstAttributDUnLieu)) AND
  PremierDansGroupeNdeN) AND AutreMembreNdeNEstUnInanime => {
    (littéraire.hiérarchie.cause Salient)
  }
  EPOUSE : PereDansLeContexte => {
    (quotidien.personne.épouse Salient)
    (quotidien.personne.partenaire_sexuel Salient)
  }
  DIVINITE : InitialeMajuscule AND ThemeMythiqueOuReligieux => {
    (religion.transcendance.divinité Salient)
  }
  BONNE : PositionNom AND AdjectifDEvaluationPositive => {
    (quotidien.attitude.humanité Salient)
```

```
    (quotidien.attitude.bienveillance Salient)
  }
  MAUVAISE : PositionNom AND AdjectifDEvaluationNegative => {
    (quotidien.attitude.humanité Negated)
    (quotidien.attitude.bienveillance Negated)
  }
  ORPHELIN : SynOrphelinPresent => {
    (quotidien.hiérarchie.génitrice Negated)
  }
  COLLECTIF : PremierDansGroupeSNdeSN AND
  AutreMembreSNdeSNEstUnCollectifDHumains => {
    (quotidien.attitude.protection Salient)
  }
  NOM : PositionNom => {
    (quotidien.hiérarchie.génitrice Salient)
    (quotidien.personne.femme Salient)
    (quotidien.attitude.bienveillance Salient)
  }
END.
```

mort

```
%%PDL
ENTRY(mort) = BEGIN
  science =
    activité =
      phénomènes_vitaux = BEGIN
        cessation;
        ralentissement;
        cessation =+=> ralentissement;
      END;
  littéraire = BEGIN
    état = BEGIN
      terminal;
      danger;
      souffrance;
      chagrin;
      épuisement;
    END;
    manière = BEGIN
      dynamisme;
      immobilité;
      peur;
      peur ===> immobilité;
      dynamisme <=/=> immobilité;
    END;
  END;
  droit =
    peine = BEGIN
      peine_capitale;
      déchéance;
    END;
  psychanalyse =
    pulsion = BEGIN
      autodestruction;
      morbidité;
      autodestruction ===> morbidité;
    END;
  religion =
    état =
      damnation;
  quotidien = BEGIN
    processus = BEGIN
```

```
      arrêt;
      extinction;
    END;
    manière = BEGIN
      mortellement;
      intensément;
      inefficace;
      hors_d_usage;
      mortellement ===> intensément;
      hors_d_usage ===> inefficace;
    END;
    état = BEGIN
      animation;
      visibilité;
      mouvement;
      pratiqué;
      absence;
      réaction;
    END;
    personne = BEGIN
      défunt;
      cadavre;
    END;
    place =
      conducteur =
        à_côté;
  END;
  jeu =
    personne =
      joueur = BEGIN
        bridge;
        quatrième;
      END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  EN_TRAIN : ((DansGroupeIntroduitParA AND PrecedeParArticleDefini) AND
  (PrecedeParDeuxDoigtsDe OR PrecedeParLArticleDe)) OR
  (DansGroupeIntroduitParSur AND (SecondDansGroupeNdeN AND
  AutreMembreNdeNEstLit)) => {
    (science.activité.phénomènes_vitaux.cessation Salient)
    (littéraire.état.terminal Salient)
  }
  ENTRE : DansGroupeEntreLaVieEtLaMort => {
    (science.activité.phénomènes_vitaux.cessation Salient)
    (littéraire.état.danger Salient)
  }
  CLINIQUE : QualifieParSynonymeDeApparent => {
    (science.activité.phénomènes_vitaux.ralentissement Salient)
  }
  SOUFFRIR : ObjetDUnSynonymeDeRessentir AND PrecedeParUnNombre => {
    (littéraire.état.souffrance Salient)
  }
  MORT_DANS_LAME : DansGroupeMortDansLAme => {
    (littéraire.état.chagrin Salient)
  }
  PEINE_CAPITALE : (SecondDansGroupeNdeN AND AutreMembreNdeNEstPeine) OR
  QualifieUnSynonymeDeCondamne => {
    (droit.peine.peine_capitale Salient)
  }
  CIVILE : QualifieParCivil => {
    (droit.peine.déchéance Salient)
  }
  RELIGION : QualifieParEternel OR QualifieParDeLAme => {
    (religion.état.damnation Salient)
  }
  PSY : ThemePsychanalyse OR (SecondDansGroupeNdeN AND
```

```
(AutreMembreNdeNEstPulsion OR AutreMembreNdeNEstTendance)) => {
  (psychanalyse.pulsion.autodestruction Salient)
}
MORT_DE : CompleteParUnSyntagmeNominalIntroduitParDe => {
  (quotidien.processus.extinction Salient)
}
BOUH : PrecedeParA AND ModeImperatif => {
  (droit.peine.peine_capitale Salient)
}
A_MORT : PrecedeParA => {
  (quotidien.manière.mortellement Salient)
  (quotidien.manière.intensément Salient)
  (droit.peine.peine_capitale Valid)
}
MORT_DE_FAIM : CompleteParUnSyntagmeNominalIntroduitParDe AND
AttributEstUneSensationPhysique => {
  (science.activité.phénomènes_vitaux.cessation Salient)
  (littéraire.état.souffrance Salient)
}
MORT_DE_PEUR : CompleteParUnSyntagmeNominalIntroduitParDe AND
AttributEstUnSynonymeDePeur => {
  (littéraire.manière.peur Salient)
}
MORT_DE_FATIGUE : CompleteParUnSyntagmeNominalIntroduitParDe AND
AttributEstUnSynonymeDeFatigue => {
  (littéraire.état.épuisement Salient)
}
MORT_OU_VIF : VifDansLeContexte => {
  (littéraire.manière.peur Salient)
  (science.activité.phénomènes_vitaux.ralentissement Salient)
}
ANGLE_MORT : PositionAdjectif AND QualifieUneRegionSpatiale => {
  (quotidien.état.visibilité Negated)
  (quotidien.état.mouvement Negated)
}
PILE_MORTE : QualifieUnInanime => {
  (quotidien.manière.hors_d_usage Salient)
}
VILLE_MORTE : PositionAdjectif => {
  (quotidien.état.animation Negated)
  (quotidien.état.mouvement Negated)
  (quotidien.état.pratiqué Negated)
}
DEFUNT : EstSujetOuObjet AND EstSupposeAnime => {
  (quotidien.personne.défunt Salient)
}
CADAVRE : EstSujetOuObjet AND EstSupposeInanime => {
  (quotidien.personne.cadavre Salient)
}
BRIDGE : VocabulaireDesJeuxDeCartes OR BridgeDansLeContexte => {
  (jeu.personne.joueur.bridge Salient)
}
PLACE_DU_MORT : SecondDansGroupeNdeN AND AutreMembreNdeNEstPlace => {
  (quotidien.place.conducteur Negated)
  (quotidien.place.conducteur.à_côté Salient)
}
FAIRE_LE_MORT : ObjetDeFaire => {
  (quotidien.état.absence Salient)
  (quotidien.état.réaction Negated)
}
IL_EST_MORT : CasNominatif => {
  (science.activité.phénomènes_vitaux.cessation Salient)
  (quotidien.personne.défunt Salient)
}
DEFAUT : _TRUE => {
  (science.activité.phénomènes_vitaux.cessation Salient)
  (quotidien.processus.arrêt Salient)
```

```
    (quotidien.personne.défunt Salient)
    (quotidien.personne.cadavre Salient)
  }
END.
```

nez

```
%%PDL
ENTRY(nez) = BEGIN
  technique = BEGIN
    organe =
      appendice_nasal;
    extrémité = BEGIN
      avant =
        fuselage;
      proue;
      cap;
      partie =
        saillante;
      avant ===> partie;
      proue ===> partie;
      cap ===> partie;
    END;
  END;
  quotidien = BEGIN
    personne =
      parfumeur;
    sens =
      odorat;
    attitude = BEGIN
      perspicace;
      haine;
      insolence;
      ivre;
      énervement;
      obéissance;
    END;
    manière = BEGIN
      approximativement;
      en_face;
      dissimulation;
    END;
    activité = BEGIN
      sortir;
      se_moquer;
      échapper_à;
      échouer;
      s_occuper_de;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  ADJ_NEZ : QualifieParAdjectifSpecialisePourNez => {
    (technique.organe.appendice_nasal Salient)
  }
  A_VUE_DE : (AutreMembreNdeNEstVue AND PremierDansGroupeNdeN) AND
  DansGroupeIntroduitParA => {
    (quotidien.manière.approximativement Salient)
  }
  IVRE : (ObjetDeAvoirEstVerre AND DansGroupeIntroduitParDans) AND
  DansComplementDeAvoir => {
    (quotidien.attitude.ivre Salient)
```

```
  }
  PIED_DE_NEZ : SecondDansGroupeNdeN AND AutreMembreNdeNEstPied => {
    (quotidien.activité.se_moquer Salient)
  }
  AVOIR_DU_NEZ : IntroduitParDu AND DansComplementDeAvoir => {
    (quotidien.sens.odorat Salient)
    (quotidien.attitude.perspicace Salient)
  }
  NEZ_FIN : QualifieParFinOuCreux => {
    (quotidien.attitude.perspicace Salient)
  }
  QQ_DANS_LE_NEZ : (DansGroupeIntroduitParDans AND ObjetDeAvoirEstIndividu)
  AND DansComplementDeAvoir => {
    (quotidien.attitude.haine Salient)
  }
  NEZ_A_NEZ : DansGroupeNezANez => {
    (quotidien.manière.en_face Salient)
  }
  ANIME : EstSupposeAnime => {
    (quotidien.personne.parfumeur Salient)
  }
  DEHORS : ComplementDuVerbeEstDehors AND ObjetDeMettre => {
    (quotidien.activité.sortir Salient)
  }
  SOUS_LE_NEZ : DansGroupeIntroduitParSous => {
    (quotidien.attitude.insolence Salient)
  }
  PASSER : DansComplementDePasser AND DansGroupeIntroduitParSous => {
    (quotidien.activité.échapper_à Salient)
  }
  AU_NEZ : SuiviParDe AND DansGroupeIntroduitParAu => {
    (quotidien.attitude.insolence Salient)
    (quotidien.manière.dissimulation Negated)
  }
  SE_CASSER : ObjetDeSeCasser => {
    (quotidien.activité.échouer Salient)
  }
  MENER : DansComplementDeMener AND DansGroupeParLeBoutDuNez => {
    (quotidien.attitude.obéissance Salient)
  }
  FOURRER : ObjetDeMettre OR ObjetDeFourrer => {
    (quotidien.activité.s_occuper_de Salient)
  }
  MONTRER : PrecedeParArticleDefini AND ObjetDeMontrer => {
    (quotidien.activité.sortir Salient)
    (quotidien.manière.dissimulation Negated)
  }
  AVION : ThemeAviation => {
    (technique.extrémité.avant.fuselage Salient)
  }
  BATEAU : ThemeMarine => {
    (technique.extrémité.proue Salient)
  }
  GEO : ThemeGeographie => {
    (technique.extrémité.cap Salient)
  }
  INANIME : ComplementDAttributionEstUnInanime => {
    (technique.extrémité.partie.saillante Salient)
  }
  MOUTARDE : SujetDuVerbeEstMoutarde AND DansComplementDeMonter => {
    (quotidien.attitude.énervement Salient)
  }
  DEFAUT : _TRUE => {
    (technique.organe.appendice_nasal Salient)
  }
END.
```

peau

```
%%PDL
ENTRY(peau) = BEGIN
  technique = BEGIN
    surface = BEGIN
      épiderme =
        organe;
      cuir;
      fourrure;
      enveloppe =
        végétal;
      croûte = BEGIN
        liquide;
        semi_liquide;
      END;
      périphérie;
      fourrure ===> cuir;
      enveloppe ===> périphérie;
      croûte ===> périphérie;
      épiderme ===> périphérie;
      épiderme <=/=> croûte;
    END;
    propriété =
      densité = BEGIN
        électricité;
        croissance;
        exponentielle;
      END;
  END;
  quotidien = BEGIN
    état = BEGIN
      état_d_esprit;
      vie;
    END;
    personne =
      individu =
        femme;
    attitude = BEGIN
      dureté;
      à_l_aise;
      conduite;
      opinion;
      amoureux;
      amoureux =+=> à_l_aise;
      dureté ===> conduite;
      amoureux <=/=> dureté;
    END;
    titre =
      injure;
    caractéristique =
      aspect;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  DANS_LA_PEAU : DansGroupeIntroduitParDans AND (ObjetDeSeMettre OR
  ObjetDeEntrerOuSynonyme) => {
    (quotidien.état.état_d_esprit Salient)
  }
  PEAU_DE_VACHE : EstSupposeAnime AND (AutreMembreNdeNEstVache AND
  PremierDansGroupeNdeN) => {
    (quotidien.personne.individu Salient)
    (quotidien.attitude.dureté Salient)
  }
```

```
  VIEILLE_PEAU : EstSupposeAnime AND QualifieParVieille => {
    (quotidien.personne.individu.femme Salient)
    (quotidien.titre.injure Salient)
  }
  BIEN : (PrecAdjPoss AND DansGroupeIntroduitParDans) AND
  ModifieAdverbePositif => {
    (quotidien.attitude.à_l_aise Salient)
  }
  MAL : (PrecAdjPoss AND DansGroupeIntroduitParDans) AND
  ModifieAdverbeNegatif => {
    (quotidien.attitude.à_l_aise Negated)
  }
  PEAU_NEUVE : QualifieParNeuve AND ObjetDeFaire => {
    (quotidien.caractéristique.aspect Salient)
    (quotidien.attitude.opinion Salient)
    (quotidien.attitude.conduite Salient)
  }
  AMOUR : (DansGroupeIntroduitParDans AND ModifieVerbeAvoir) AND
  DansGroupeAdverbial => {
    (quotidien.attitude.amoureux Salient)
  }
  VIE : PrecAdjPoss AND ((ObjetDeRisquer OR ObjetDeVendre) OR
  ObjetDeTenirA) => {
    (quotidien.état.vie Salient)
  }
  TUER : PrecedeParArticleDefini AND ObjetDeFaire => {
    (quotidien.état.vie Negated)
  }
  ANIMAL : AutreMembreSNdeSNEstUnAnimal AND PremierDansGroupeSNdeSN => {
    (technique.surface.cuir Salient)
    (technique.surface.fourrure Salient)
  }
  VEGETAL : AutreMembreSNdeSNEstUnVegetal AND PremierDansGroupeSNdeSN => {
    (technique.surface.enveloppe.végétal Salient)
  }
  SOLIDE : AutreMembreSNdeSNEstUnSolide AND PremierDansGroupeSNdeSN => {
    (technique.surface.périphérie Salient)
  }
  LIQUIDE : AutreMembreSNdeSNEstUnLiquide AND PremierDansGroupeSNdeSN => {
    (technique.surface.croûte.liquide Salient)
  }
  SEMI_LIQUIDE : AutreMembreSNdeSNEstUnSemiLiquide AND
  PremierDansGroupeSNdeSN => {
    (technique.surface.croûte.semi_liquide Salient)
  }
  EFFET_DE_PEAU : AutreMembreNdeNEstEffet AND SecondDansGroupeNdeN => {
    (technique.surface.périphérie Salient)
    (technique.propriété.densité.électricité Salient)
  }
  DEFAUT : _TRUE => {
    (technique.surface.épiderme Salient)
  }
END.
```

père

```
%%PDL
ENTRY(père) = BEGIN
  quotidien = BEGIN
    titre =
      monsieur = BEGIN
        péjoratif;
        âge;
```

```
      END;
    hiérarchie = BEGIN
      géniteur =
        famille;
      générateur;
      antécédant;
      similaire;
      responsable = BEGIN
        sûr;
        modeste;
      END;
      géniteur ===> générateur;
      générateur ===> antécédant;
      générateur ===> similaire;
      géniteur ===> responsable;
    END;
    attitude = BEGIN
      guide = BEGIN
        spirituel;
        conscience;
        évolution;
      END;
      bienveillance;
      éducateur;
      protecteur;
      éducateur =-=> bienveillance;
      guide ===> éducateur;
      bienveillance ===> protecteur;
    END;
  END;
  théâtre =
    attitude = BEGIN
      grave;
      digne;
    END;
  religion =
    personne = BEGIN
      dieu = BEGIN
        éternel;
        trinité;
      END;
      pape =
        saint;
      théologien = BEGIN
        église;
        antiquité;
        écrivain;
      END;
      prêtre;
    END;
  littéraire =
    hiérarchie = BEGIN
      ancêtre;
      fondateur;
    END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  DE_FAMILLE : SuiviParDeFamille => {
    (quotidien.hiérarchie.géniteur.famille Salient)
    (quotidien.hiérarchie.responsable Salient)
  }
  SAINT_PERE : PrecedeParSaint => {
    (religion.personne.pape.saint Salient)
  }
  FONDATEUR : CompleteParUnInanimeAuGenitif => {
    (littéraire.hiérarchie.fondateur Salient)
```

```
  }
  ANCETRES : Pluriel AND PrecedeParPronomPossessif => {
    (littéraire.hiérarchie.ancêtre Salient)
  }
  DIEU_LE_PERE : DieuLePere => {
    (religion.personne.dieu Salient)
  }
  PERE_ETERNEL : SuiviParEternel => {
    (religion.personne.dieu.éternel Salient)
  }
  TRINITE : InitialeMajuscule AND (SaintEspritDansLeContexte OR
  FilsMajusculeDansLeContexte) => {
    (religion.personne.dieu.trinité Salient)
  }
  PERE_DE_L_EGLISE : SuiviParDeLEglise => {
    (religion.personne.théologien.église Salient)
  }
  PERE_DU_DESERT : SuiviParDuDesert => {
    (religion.personne.théologien Salient)
  }
  PRETRE : PrecedeParBon => {
    (religion.personne.prêtre Salient)
  }
  THEATRE : VocabulaireTheatre => {
    (théâtre.attitude Salient)
  }
  LE_PERE_X : PrecedeParArticleDefini AND SuiviParNomPropre => {
    (quotidien.titre.monsieur Salient)
    (religion.personne.prêtre Salient)
  }
  FILS1 : FilsDansLeContexte AND PositionNom => {
    (quotidien.hiérarchie.géniteur Salient)
  }
  FILS2 : PositionAdjectif AND FilsDansLeContexte => {
    (quotidien.hiérarchie.générateur Salient)
  }
  PUNIR : PunirDansLeContexte => {
    (quotidien.attitude.éducateur Salient)
  }
  SPIRITUEL : SuiviParSpirituel => {
    (quotidien.attitude.guide.spirituel Salient)
  }
  POUR : PourPronomDansLeContexte => {
    (quotidien.attitude.guide Salient)
    (quotidien.attitude.bienveillance Salient)
  }
  NEG : PrecedeParNegation AND EstUnAttribut => {
    (quotidien.hiérarchie.géniteur Negated)
  }
  ATTRIBUT : EstUnAttribut => {
    (quotidien.hiérarchie.responsable Salient)
    (quotidien.attitude.bienveillance Salient)
  }
  DEFAUT : _TRUE => {
    (quotidien.hiérarchie.géniteur Valid)
    (quotidien.hiérarchie.responsable Valid)
    (quotidien.attitude.bienveillance Valid)
  }
END.
```

roi

```
%%PDL
```

```
ENTRY(roi) = BEGIN
  littéraire =
    titre =
      monarque = BEGIN
        france;
        espagne;
        perse;
        espagne <=/=> france;
        espagne <=/=> perse;
        france <=/=> perse;
      END;
  politique =
    titre =
      souverain;
  quotidien =
    supériorité = BEGIN
      principal;
      personne =
        superlatif;
      lion;
    END;
  jeu =
    GHOST = BEGIN
      pièce = BEGIN
        échecs;
        principal;
      END;
      carte = BEGIN
        coeur;
        carreau;
        trèfle;
        pique;
      END;
    END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  PERSE : (InitialeMajuscule AND PrecedeParArticleDefini) AND PrecedeParGrand => {
    (littéraire.titre.monarque.perse Salient)
  }
  FRANCE : (InitialeMajuscule AND PrecedeParArticleDefini) AND SuiviParTresChretien => {
    (littéraire.titre.monarque.france Salient)
  }
  ESPAGNE : (InitialeMajuscule AND PrecedeParArticleDefini) AND SuiviParCatholique => {
    (littéraire.titre.monarque.espagne Salient)
  }
  SOUVERAIN : PremierDansGroupeNdeN AND (AutreMembreNdeNEstPays OR
  AutreMembreNdeNEstSubstantifDeNationalite) => {
    (politique.titre.souverain Salient)
  }
  LION : PremierDansGroupeNdeN AND AutreMembreNdeNEstAnimaux => {
    (quotidien.supériorité.lion Salient)
  }
  SUPER : PremierDansGroupeNdeN AND AutreMembreNdeNDenoteUneCategorie => {
    (quotidien.supériorité.personne.superlatif Salient)
  }
  PRINCIPAL : PremierDansGroupeNdeN AND AutreMembreNdeNEstUnInanime => {
    (quotidien.supériorité.principal Salient)
  }
  ECHECS : VocabulaireDesEchecs => {
    (jeu.GHOST.pièce.échecs Salient)
  }
  COEUR : PremierDansGroupeNdeN AND AutreMembreNdeNEstCoeur => {
    (jeu.GHOST.carte.coeur Salient)
  }
  CARREAU : PremierDansGroupeNdeN AND AutreMembreNdeNEstCarreau => {
    (jeu.GHOST.carte.carreau Salient)
```

```
  }
  PIQUE : PremierDansGroupeNdeN AND AutreMembreNdeNEstPique => {
    (jeu.GHOST.carte.pique Salient)
  }
  TREFLE : PremierDansGroupeNdeN AND AutreMembreNdeNEstTrefle => {
    (jeu.GHOST.carte.trèfle Salient)
  }
  CARTES : VocabulaireDesJeuxDeCartes => {
    (jeu.GHOST.carte Salient)
  }
  DEFAUT : _TRUE => {
    (politique.titre.souverain Salient)
  }
END.
```

soeur

```
%%PDL
ENTRY(soeur) = BEGIN
  religion = BEGIN
    personne =
      religieuse;
    titre =
      religieuse;
  END;
  quotidien = BEGIN
    personne =
      femme = BEGIN
        famille;
        fratrie;
        féminin;
        fratrie ===> famille;
      END;
    hiérarchie = BEGIN
      similaire;
      simultané;
      apparenté;
      simultané ===> similaire;
      apparenté ===> similaire;
    END;
    sentiment = BEGIN
      compagnon;
      optimal;
      proche;
      optimal ===> compagnon;
      compagnon ===> proche;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  VOCATIF : CasVocatif => {
    (religion.personne.religieuse Salient)
    (religion.titre.religieuse Salient)
  }
  MA_SOEUR : PositionNom AND PrecAdjPoss => {
    (quotidien.personne.femme.fratrie Salient)
    (quotidien.sentiment.proche Salient)
  }
  SOEUR_DE_X : PositionNom AND CompleteParNPAuGenitif => {
    (quotidien.personne.femme.fratrie Salient)
  }
  BONNE_SOEUR : PositionNom AND QualifieParBonne => {
```

```
    (religion.personne.religieuse Salient)
  }
  SOEUR_DE_SANG : PositionNom AND (PremierDansGroupeSNdeSN AND
  AutreMembreSNdeSNEstUnInanime) => {
    (quotidien.sentiment.compagnon Salient)
  }
  ADJECTIF : PositionAdjectif => {
    (quotidien.hiérarchie.apparenté Salient)
    (quotidien.hiérarchie.simultané Salient)
  }
  NOM : PositionNom => {
    (quotidien.personne.femme.fratrie Salient)
  }
  AME_SOEUR : PositionAdjectif AND QualifieAme => {
    (quotidien.sentiment.optimal Salient)
  }
END.
```

soleil

```
%%PDL
ENTRY(soleil) = BEGIN
  quotidien =
    propriété_physique = BEGIN
      lumière;
      chaleur;
      beau_temps;
      beau_temps ===> chaleur;
      beau_temps ===> lumière;
    END;
  astronomie =
    objet = BEGIN
      le_soleil;
      étoile;
      le_soleil =-=> étoile;
    END;
  littéraire = BEGIN
    lieu =
      orient;
    couleur =
      jaune = BEGIN
        gai;
        lumineux;
        brûlant;
      END;
    statut = BEGIN
      puissance =
        superlatif;
      gloire;
      dirigeant;
    END;
    transcendance = BEGIN
      centre;
      dieu;
      source_de_vie;
      dieu ===> centre;
      dieu ===> source_de_vie;
    END;
  END;
  politique =
    nation =
      japon;
  botanique =
```

```
    espèce =
      tournesol;
  sport =
    mouvement =
      tour_complet;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  SPORT : ThemeSport OR (VerbeDeMouvement AND SujetAnime) => {
    (sport.mouvement.tour_complet Salient)
    (botanique.espèce.tournesol Ignored)
    (politique.nation.japon Ignored)
    (astronomie Ignored)
  }
  FLEUR : ThemeBotanique => {
    (botanique.espèce.tournesol Salient)
    (sport.mouvement.tour_complet Ignored)
    (astronomie Ignored)
  }
  JAPON : ComplementDeEmpire AND QualifieParLevant => {
    (politique.nation.japon Salient)
    (littéraire.lieu.orient Salient)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  AU_SOLEIL : (CasLocatif AND PrecedeParAu) OR (NOT(ThemeSport) AND VerbeDeMouvement) => {
    (quotidien.propriété_physique.lumière Salient)
    (quotidien.propriété_physique.chaleur Salient)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
    (politique.nation.japon Ignored)
  }
  PAS_DE_SOLEIL : (CasExistentiel OR LexiqueMeteo) AND PrecedeParNegation => {
    (quotidien.propriété_physique.beau_temps Negated)
    (politique.nation.japon Ignored)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  DU_SOLEIL : (CasExistentiel AND PrecedeParUnQuantitatif) OR LexiqueMeteo => {
    (quotidien.propriété_physique.beau_temps Salient)
    (politique.nation.japon Ignored)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  LE_SOLEIL : (InitialeMajuscule AND PrecedeParArticleDefini) OR
  (CasNominatif AND VerbeSupposeSujetAnime) => {
    (astronomie.objet.le_soleil Salient)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  UN_SOLEIL : ThemeAstronomie => {
    (astronomie.objet.étoile Salient)
    (politique.nation.japon Ignored)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  QUALIFIE_ANIME : PositionAdjectif AND QualifieUnAnime => {
    (littéraire.statut.puissance.superlatif Salient)
    (littéraire.statut.dirigeant Salient)
    (littéraire.transcendance.centre Salient)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
  }
  QUALIFIE_INANIME : PositionAdjectif AND QualifieUnInanime => {
    (littéraire.couleur.jaune Salient)
    (quotidien.propriété_physique.lumière Salient)
  }
```

```
  DEFAUT : PositionNom AND (CasNominatif OR CasAccusatif) => {
    (littéraire.statut.puissance.superlatif Salient)
    (littéraire.statut.gloire Salient)
    (littéraire.transcendance.centre Salient)
    (littéraire.transcendance.source_de_vie Salient)
  }
  DIVINITE : (PositionNom AND PrecedeParArticleDefini) AND ThemeMythiqueOuReligieux => {
    (astronomie.objet.le_soleil Salient)
    (littéraire.statut.puissance Salient)
    (littéraire.transcendance.dieu Salient)
    (botanique.espèce.tournesol Ignored)
    (sport.mouvement.tour_complet Ignored)
    (politique.nation.japon Ignored)
  }
END.
```

système

```
%%PDL
ENTRY(système) = BEGIN
  littéraire =
    ensemble =
      ensemble_d_idées = BEGIN
        scientifique;
        philosophique;
      END;
  informatique = BEGIN
    processus = BEGIN
      système_d_exploitation =
        appartenant_à;
      logiciels;
    END;
    combinaison = BEGIN
      architecture_logicielle =
        distribué;
      logiciels;
    END;
    traitement =
      système_expert =
        intelligence_artificielle;
  END;
  psychanalyse =
    combinaison =
      instance;
  finance =
    combinaison =
      système_monétaire;
  quotidien = BEGIN
    fonction =
      méthodes;
    finalité =
      moyen =
        habile;
    combinaison =
      équilibre_émotionnel;
    artefact =
      dispositif =
        fonction_déterminée;
  END;
  politique =
    organisation = BEGIN
      gouvernement;
      administration;
```

```
      fonctionnement;
    END;
  science =
    combinaison = BEGIN
      ensemble_d_unités =
        fonctionnement_global;
      organes =
        même_nature;
    END;
  linguistique =
    combinaison =
      unités = BEGIN
        définitions_réciproques;
        minimal;
        phonologique;
        sémantique;
      END;
  maths =
    combinaison =
      critères = BEGIN
        équations;
        contraintes;
      END;
  physique = BEGIN
    combinaison =
      référentiel = BEGIN
        cinétique;
        point_matériel;
      END;
    définition =
      système_international =
        unités;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  MONETAIRE : AdjectifEstMonetaire OR DomaineFinance => {
    (finance.combinaison.système_monétaire Salient)
  }
  PHILO : AdjectifEstPhilosophique OR DomainePhilosophie => {
    (littéraire.ensemble.ensemble_d_idées.philosophique Salient)
  }
  SCIENCE : DomaineScience => {
    (littéraire.ensemble.ensemble_d_idées.scientifique Salient)
  }
  PSY : DomainePsychanalyse => {
    (psychanalyse.combinaison.instance Salient)
  }
  ANATOMIE : PositionNom AND AdjectifMedical => {
    (science.combinaison.organes Salient)
  }
  SI : DomaineScience AND (PositionNom AND AdjectifEstInternational) => {
    (science.combinaison.ensemble_d_unités Salient)
    (physique.définition.système_international Salient)
  }
  COURRIR : PrecArticleDef AND ObjetDeCourrirSur => {
    (quotidien.combinaison.équilibre_émotionnel Negated)
  }
  CONTRAINTES : DomaineMaths AND ContrainteDansLeContexte => {
    (maths.combinaison.critères.contraintes Salient)
  }
  EQUATIONS : DomaineMaths => {
    (maths.combinaison.critères.équations Salient)
  }
  REFERENCE : (QualifieParReferentiel OR QualifieParDeReference) OR
  (ThemeEstPointMateriel AND DomaineScience) => {
    (physique.combinaison.référentiel Salient)
```

```
  }
  ECO : AdjectifEconomique AND PositionNom => {
    (politique.organisation.fonctionnement Salient)
  }
  POLITIQUE : AdjectifPolitique AND PositionNom => {
    (politique.organisation.gouvernement Salient)
  }
  ADJ_SYST : PositionAdjectif AND DomaineInformatique => {
    (informatique.processus.système_d_exploitation.appartenant_à Salient)
  }
  SE : (AdjectifEstExpert OR ThemeIntelligenceArtificielle) AND DomaineInformatique => {
    (informatique.traitement.système_expert Salient)
  }
  LE_SYSTEME : (PasAdjectif AND PrecArticleDef) AND DomaineInformatique => {
    (informatique.processus.système_d_exploitation Salient)
    (informatique.combinaison.logiciels Salient)
    (informatique.processus.logiciels Salient)
  }
  DISTRIB : AdjectifEstDistribue AND DomaineInformatique => {
    (informatique.combinaison.architecture_logicielle.distribué Salient)
  }
  DEFAUT : _TRUE => {
    (quotidien.fonction.méthodes Salient)
    (quotidien.finalité.moyen Salient)
    (quotidien.artefact.dispositif Salient)
  }
END.
```

toucher

```
%%PDL
ENTRY(toucher) = BEGIN
  médecine =
    pratique =
      examen;
  musique =
    propriété = BEGIN
      caractère;
      style;
      caractère ===> style;
      style ===> caractère;
    END;
  technique = BEGIN
    perception = BEGIN
      sens;
      impression;
      impression ===> sens;
    END;
    processus =
      contact;
    état = BEGIN
      contact;
      contigu;
      contigu ===> contact;
    END;
  END;
  quotidien = BEGIN
    processus = BEGIN
      mouvement = BEGIN
        contact;
        main;
      END;
      atteindre;
```

```
      concerner;
      percevoir;
      aborder;
      modifier;
      consommer;
      émouvoir;
      aborder ===> atteindre;
      consommer ===> modifier;
      atteindre =-=> modifier;
      atteindre =-=> consommer;
      concerner =-=> atteindre;
      concerner =-=> consommer;
      concerner =-=> aborder;
    END;
    expression = BEGIN
      communiquer;
      dire;
      dissimuler;
      dire ===> communiquer;
      dissimuler <=/=> dire;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  IMPERATIF : ModeImperatif => {
    (technique.processus.contact Salient)
    (quotidien.processus.mouvement.main Salient)
  }
  ANIME : AttribueAUnAnime AND PositionNom => {
    (musique.propriété.style Salient)
  }
  INANIME : AttribueAUnInanime AND PositionNom => {
    (technique.perception.impression Salient)
  }
  MEDICAL : PositionNom AND AdjectifMedical => {
    (médecine.pratique.examen Salient)
  }
  NOM : PositionNom => {
    (technique.perception.sens Salient)
  }
  ARGENT : TransitifDirect AND ObjetSynInfDeRetribution => {
    (quotidien.processus.percevoir Salient)
  }
  TOUCHER_UN_MOT : TransitifDirect AND ObjetContientMot => {
    (quotidien.expression.dire Salient)
  }
  BLESSER : TransitifDirect AND SujetOuObjetDansVocabulaireGuerrier => {
    (quotidien.processus.atteindre Salient)
  }
  MEDIA : TransitifDirect AND MediaDansLeContexte => {
    (quotidien.expression.communiquer Salient)
  }
  CONCERNER : ObjetEstUnCollectifOuUneInstitution => {
    (quotidien.processus.concerner Salient)
  }
  EMOUVOIR : ObjetEstUnePersonne AND TransitifDirect => {
    (quotidien.processus.émouvoir Salient)
  }
  IND_INANIME : TransitifIndirect AND ObjetEstUnInanimeConcret => {
    (technique.état.contact Salient)
    (technique.état.contigu Salient)
    (quotidien.processus.mouvement.contact Salient)
  }
  ALIMENT : TransitifIndirect AND ObjetEstUnAliment => {
    (quotidien.processus.consommer Salient)
  }
```

```
  LIEU : TransitifIndirect AND ObjetEstUnLieu => {
    (quotidien.processus.aborder Salient)
  }
  PRODUIT : TransitifIndirect AND ObjetEstUnProduit => {
    (quotidien.processus.modifier Salient)
  }
  GENERAL : _TRUE => {
    (technique.processus.contact Salient)
    (technique.état.contact Salient)
    (quotidien.processus.mouvement.contact Salient)
  }
END.
```

vie

```
%%PDL
ENTRY(vie) = BEGIN
  science =
    activité = BEGIN
      phénomènes_vitaux = BEGIN
        nutrition;
        assimilation;
        croissance;
        reproduction;
      END;
      existence = BEGIN
        naissance;
        mort;
      END;
      phénomènes_vitaux =+=> existence;
    END;
  religion =
    état =
      divinité = BEGIN
        bonheur;
        paradis;
        mort;
      END;
  littéraire = BEGIN
    manière = BEGIN
      entrain;
      dynamisme;
      entrain ===> dynamisme;
    END;
    oeuvre =
      biographie;
    activité =
      existence = BEGIN
        naissance;
        mort;
        évolution;
      END;
  END;
  quotidien = BEGIN
    durée = BEGIN
      existence = BEGIN
        naissance;
        mort;
      END;
      jamais;
    END;
    événement = BEGIN
      expérience;
```

```
      organisation;
      activité;
      plaisir =
        excès;
      expérience ===> organisation;
      activité =-=> plaisir;
    END;
    manière = BEGIN
      nullement;
      mode_de_vie;
      insupportable;
      mode_de_vie =-=> nullement;
      nullement =-=> mode_de_vie;
    END;
    condition = BEGIN
      moyen_matériel = BEGIN
        aliment;
        argent;
        argent ===> aliment;
      END;
      condition_humaine;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  DEVOIR : ObjetDeDevoir => {
    (science.activité.existence Salient)
  }
  DONNER : ObjetDeDonner => {
    (science.activité.existence.naissance Salient)
  }
  MOURIR : ObjetDePrendre OR ObjetDePerdre => {
    (science.activité.existence.mort Salient)
  }
  ETERNELLE : QualifieParEternel => {
    (religion.état.divinité.bonheur Salient)
  }
  DEBORDER : SecondDansGroupeVdeN AND AutreMembreVdeNEstDeborderOuEclater => {
    (littéraire.manière.entrain Salient)
  }
  DUREE : QualifieParAdjectifDeDuree => {
    (quotidien.durée.existence Salient)
  }
  REUSSIR : ObjetDeReussir => {
    (quotidien.durée.existence Salient)
    (quotidien.manière.mode_de_vie Salient)
  }
  RATER : ObjetDeRater => {
    (quotidien.durée.existence Salient)
    (quotidien.manière.mode_de_vie Negated)
  }
  REFAIRE : ObjetDeRefaire => {
    (quotidien.événement.organisation Salient)
  }
  A_VIE : PrecedeParA OR DansGroupeALaVieALaMort => {
    (quotidien.durée.existence Salient)
    (quotidien.durée.jamais Negated)
  }
  JAMAIS : DansGroupeJamaisDeLaVie => {
    (quotidien.manière.nullement Salient)
  }
  MENER : ObjetDeMener => {
    (quotidien.manière.mode_de_vie Salient)
    (quotidien.événement.expérience Salient)
  }
  PAS_UNE_VIE : (PrecedeParNegation AND EstUnAttribut) AND PrecedeParArticleIndefini => {
```

```
    (quotidien.manière.insupportable Salient)
  }
  ACTIVITE : QualifieParAdjectifDeriveDeDomaineDActivite => {
    (quotidien.événement.activité Salient)
  }
  PAS_CHER : QualifieParEvaluationDePrixPositive => {
    (quotidien.condition.moyen_matériel.argent Salient)
  }
  CHER : QualifieParEvaluationDePrixNegative => {
    (quotidien.condition.moyen_matériel.argent Negated)
  }
  DUR : QualifieParDure => {
    (quotidien.manière.mode_de_vie Negated)
    (quotidien.condition.condition_humaine Negated)
  }
  DOUX : QualifieParDouce => {
    (quotidien.manière.mode_de_vie Salient)
    (quotidien.condition.condition_humaine Salient)
  }
  GAGNER : ObjetDeGagner => {
    (quotidien.condition.moyen_matériel.argent Salient)
  }
  FAIRE : ObjetDeFaire => {
    (quotidien.événement.plaisir.excès Salient)
    (quotidien.manière.insupportable Salient)
  }
  LA_VIE : ObjetDeConnaitreOuAffronter => {
    (quotidien.condition.condition_humaine Salient)
  }
  BIOGRAPHIE : CompleteParNPAuGenitif => {
    (littéraire.oeuvre.biographie Salient)
  }
  VIE_DE_CHIEN : PremierDansGroupeNdeN AND AutreMembreNdeNEstChien => {
    (quotidien.manière.mode_de_vie Negated)
    (quotidien.condition.condition_humaine Negated)
  }
  ADJ : DansSyntagmeAdjectival => {
    (littéraire.manière.dynamisme Salient)
  }
  EVOLUTION : PremierDansGroupeNdeN AND AutreMembreNdeNEstInanime => {
    (quotidien.événement.expérience Salient)
    (quotidien.durée.existence Salient)
  }
  DEFAUT : _TRUE => {
    (quotidien.événement.expérience Salient)
    (quotidien.durée.existence Salient)
  }
END.
```

vieux

```
%%PDL
ENTRY(vieux) = BEGIN
  technique = BEGIN
    état = BEGIN
      âgé;
      ancien;
      ancien ===> âgé;
    END;
    processus =
      vieillir;
  END;
  quotidien = BEGIN
```

```
    processus =
      vivre;
    manière = BEGIN
      longtemps;
      brusquement;
    END;
    état = BEGIN
      vieillesse;
      usé;
      surrané;
      surrané ===> usé;
    END;
    personne = BEGIN
      individu;
      parent;
      ami;
      vétéran;
      vétéran ===> individu;
      parent ===> individu;
      ami ===> individu;
      parent <=/=> ami;
    END;
    attitude = BEGIN
      vigoureux;
      charmant;
    END;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  MON_VIEUX1 : CasVocatif AND PrecAdjPoss => {
    (quotidien.personne.ami Salient)
  }
  MON_VIEUX2 : NOT(CasVocatif) AND PrecAdjPoss => {
    (quotidien.personne.parent Salient)
  }
  SE_FAIRE_VIEUX : ObjetDeSeFaire => {
    (technique.processus.vieillir Salient)
  }
  VETERAN : AutreMembreSNdeSNEstLaVieille AND PremierDansGroupeSNdeSN => {
    (quotidien.personne.vétéran Salient)
  }
  VIEUX_JOURS : QualifieJours => {
    (quotidien.état.vieillesse Salient)
  }
  VIEUX_OS1 : QualifieOs AND ObjetDeFaire => {
    (quotidien.processus.vivre Salient)
    (quotidien.manière.longtemps Salient)
  }
  VIEUX_OS2 : (PrecedeParNegation AND QualifieOs) AND ObjetDeFaire => {
    (quotidien.processus.vivre Salient)
    (quotidien.manière.longtemps Negated)
  }
  VIEUX_JEU : PasArticle AND QualifieJeu => {
    (quotidien.état.surrané Salient)
  }
  INANIME : QualifieUnInanime => {
    (technique.état.ancien Salient)
    (quotidien.état.usé Salient)
  }
  COUP : AutreMembreNdeNEstCoup AND SecondDansGroupeNdeN => {
    (technique.processus.vieillir Salient)
    (quotidien.manière.brusquement Salient)
  }
  ANIME : QualifieUnAnime => {
    (technique.état.âgé Salient)
    (technique.état.ancien Salient)
```

```
    (quotidien.manière.longtemps Salient)
    (quotidien.attitude.vigoureux Negated)
    (quotidien.attitude.charmant Negated)
  }
  NOM : PositionNom => {
    (technique.état.âgé Salient)
    (quotidien.personne.individu Salient)
  }
END.
```

voir

```
%%PDL
ENTRY(voir) = BEGIN
  technique = BEGIN
    processus = BEGIN
      percevoir =
        yeux;
      regarder;
      regarder ===> percevoir;
    END;
    manière =
      attentivement;
  END;
  littéraire = BEGIN
    événement = BEGIN
      naissance;
      apparition;
      naissance ===> apparition;
    END;
    processus = BEGIN
      assister_à;
      être_témoin_de;
      rencontrer;
      visiter;
      se_représenter;
      juger;
      veiller_à;
      assister_à ===> être_témoin_de;
      être_témoin_de ===> assister_à;
      juger ===> se_représenter;
    END;
    manière = BEGIN
      fréquemment;
      mentalement;
    END;
  END;
  quotidien = BEGIN
    processus = BEGIN
      montrer;
      s_éloigner;
      causer_des_ennuis;
      apprécier;
      concevoir;
      prévoir;
      essayer;
      se_produire;
      attendre;
      prévoir ===> concevoir;
      apprécier ===> concevoir;
      montrer =-=> causer_des_ennuis;
      montrer =-=> s_éloigner;
      essayer ===> prévoir;
```

```
      se_produire =-=> causer_des_ennuis;
      se_produire =-=> s_éloigner;
    END;
    manière = BEGIN
      visiblement;
      péjoratif;
      perspicace;
      en_rapport;
      péjoratif =-=> perspicace;
      perspicace =-=> péjoratif;
    END;
    interjection =
      rappel_à_l_ordre;
  END;
LIBRARY
  D:\LightPeleas\Predicats.dll;
RULES
  ALLER_SE_FAIRE_VOIR : ((DependDeAller AND PremierVerbeEstFaire) AND
  DeuxiemeVerbe) AND PlusieursVerbes => {
    (quotidien.processus.s_éloigner Salient)
    (quotidien.manière.péjoratif Salient)
    (quotidien.processus.montrer Ignored)
    (quotidien.interjection.rappel_à_l_ordre Ignored)
  }
  FAIRE_VOIR : (PremierVerbeEstFaire AND DeuxiemeVerbe) AND PlusieursVerbes => {
    (quotidien.processus.montrer Salient)
    (quotidien.manière.visiblement Salient)
  }
  VOIR_LE_JOUR1 : SujetAnime AND ObjetEstLeJour => {
    (littéraire.événement.naissance Salient)
  }
  VOIR_LE_JOUR2 : ObjetEstLeJour AND (SujetEstUnInanimeConcret OR
  SujetEstUnInanimeAbstrait) => {
    (littéraire.événement.apparition Salient)
  }
  EN_COULEURS : AdverbeContientCouleurs AND ObjetEstEn => {
    (quotidien.processus.causer_des_ennuis Salient)
  }
  ASSISTER : ObjetEstUnEvenement => {
    (littéraire.processus.assister_à Salient)
  }
  REGARDER : ModeImperatif => {
    (technique.processus.regarder Salient)
    (technique.manière.attentivement Salient)
  }
  VOCATIF : CasVocatif => {
    (quotidien.interjection.rappel_à_l_ordre Salient)
  }
  FREQUENTER : ObjetEstUnePersonne => {
    (littéraire.processus.rencontrer Salient)
    (littéraire.manière.fréquemment Salient)
  }
  VISITER : ObjetEstUnLieu => {
    (littéraire.processus.visiter Salient)
  }
  ABSTRAIT : ObjetEstAbstrait => {
    (littéraire.processus.se_représenter Salient)
    (littéraire.manière.mentalement Salient)
    (quotidien.processus.concevoir Salient)
  }
  BIEN : AdverbeContientBien => {
    (littéraire.manière.mentalement Salient)
    (littéraire.processus.se_représenter Salient)
  }
  MAL : PrecedeParNegation AND AdverbeContientBien => {
    (littéraire.processus.se_représenter Negated)
    (littéraire.manière.mentalement Salient)
```

```
  }
  VOIR_LOIN : AdverbeEstLoin => {
    (quotidien.processus.prévoir Salient)
    (quotidien.manière.perspicace Salient)
  }
  VOIR_VENIR : ((PasDObjet AND DeuxiemeVerbeEstVenir) AND PremierVerbe)
  AND PlusieursVerbes => {
    (quotidien.processus.attendre Salient)
  }
  TE_VOIR_VENIR : ((NOT(PasDObjet) AND DeuxiemeVerbeEstVenir) AND
  PremierVerbe) AND PlusieursVerbes => {
    (quotidien.processus.prévoir Salient)
    (quotidien.manière.perspicace Salient)
  }
  JUGER : (ObjetEstGenerique AND DeuxiemeVerbe) AND PasDObjet => {
    (littéraire.processus.juger Salient)
  }
  POUR_VOIR : IntroduitParPour => {
    (quotidien.processus.essayer Salient)
  }
  ESSAYER : IntroduitParPour => {
    (quotidien.processus.essayer Salient)
  }
  A_VOIR1 : PasDObjet AND DansGroupeIntroduitParA => {
    (quotidien.processus.essayer Salient)
    (littéraire.processus.se_représenter Salient)
    (littéraire.manière.mentalement Salient)
  }
  A_VOIR2 : SAppliqueAUnEnsembleNonVide AND DansGroupeIntroduitParA => {
    (quotidien.manière.en_rapport Salient)
  }
  A_VOIR3 : SAppliqueAUnEnsembleVide AND DansGroupeIntroduitParA => {
    (quotidien.manière.en_rapport Negated)
  }
  VEILLER : ObjetEstUnProcessus AND TransitifIndirect => {
    (littéraire.processus.veiller_à Salient)
  }
  SE_PRODUIRE : (SujetEstUnInanimeAbstrait OR SujetEstGenerique) AND
  VerbePronominal => {
    (quotidien.processus.se_produire Salient)
  }
  OEIL1 : AdjectifDEvaluationPositive AND AdverbeContientOeil => {
    (quotidien.processus.apprécier Salient)
  }
  OEIL2 : AdverbeContientOeil AND AdjectifDEvaluationNegative => {
    (quotidien.processus.apprécier Negated)
  }
  GENERAL : _TRUE => {
    (technique.processus.percevoir Salient)
    (littéraire.processus.se_représenter Salient)
    (littéraire.manière.mentalement Salient)
    (quotidien.manière.visiblement Salient)
  }
END.
```

A.2 Corpus de test

Le corpus de phrases sur lequel PELEAS a été testé comporte 110 phrases qui sont présentées ci-dessous. Les phrases avec une astérisque ont déjà été testées sur le modèle EDGAR.

1. Pour le mot *bouche* :

(a) *Dans sa **bouche**, les mots prennent un ton plus subtil.*
Isotopie entre *bouche* et *mots*.

(b) *Les pompiers font la fine **bouche**.*
Superposition d'un emploi attesté (*fine bouche*) et d'une isotopie avec *pompiers*.

(c) *Elle porte un doigt à sa **bouche**.*
Absence d'indice contextuel.

(d) *La famine chassait les hommes de la montagne, effrayés par le poids des **bouches** à nourrir.*
Présence d'un emploi attesté.

(e) *En psychanalyse, Gallimard publie la **Bouche** de l'inconscient.*
Emploi catachrétique.

(f) *Les textes de Brillat-Savarin font venir l'eau aux fines **bouches**.*
Superposition de deux emplois attestés.

(g) *Clyde déposa un baiser sur la **bouche** tiède de son colt.*
Collision de deux isotopies.

2. Pour le mot *fille* :

(a) * *C'est une bonne **fille**.*
Gestion de la polysémie d'acception entre 'enfant féminin' et 'individu féminin'.

(b) * *La cellule mère produit deux cellules **filles** identiques.*
Emploi adjectival particulier doublé d'une relation mère / fille.

(c) *C'était une bonne **fille** mais publique.*
Collisition entre une isotopie et un emploi attesté.

(d) *La **fille** de salle l'était restée.*
Superposition de deux emplois attestés.

3. Pour le mot *fils* :

(a) *Tel père, tel **fils** !*
Lexie figée.

(b) * *Le processus **fils** est une image du processus père.*
Emploi adjectival particulier doublé d'une relation père / fils.

(c) *C'est un bon **fils** à papa.*
Superposition et collision entre un emploi attesté et une isotopie.

(d) *Il faut préserver l'honneur des* Fils *de France.*
Gestion de la majuscule en-dehors du contexte religieux.

4. Pour le mot *frère* :

(a) *Le pire parjure est d'être un faux **frère** de sang.*
Collision entre deux emplois attestés.

(b) * *C'est un **frère** pour moi.*
Gestion d'une analogie / abstraction.

(c) *La France et l'Allemagne sont deux pays **frères**.*
Emploi adjectival.

(d) *Délivrez moi ô mes **frères** !*
Gestion du vocatif.

5. Pour le mot *jeune* :

(a) *Ca se passe à la maison des **jeunes**.*
Emploi nominal particulier.

(b) *Ce vin est un peu **jeune**.*
Qualification d'un inanimé dans le domaine particulier de l'œnologie.

(c) *Je me suis senti un peu **jeune** sur ce projet.*
Superposition d'emplois due à une isotopie issue de *projet* (en tant qu'activité).

(d) *C'est **jeune**, un **jeune**...*
Structure tautologique où le premier emploi est adjectival et le second est nominal.

6. Pour le mot *langue* :

(a) *Le peuple tirait la **langue**.*
Emploi particulier de *tirer la langue* lorsque le sujet n'est pas un individu.

(b) *Il étouffe car il a avalé sa **langue**.*
Possibilité de rencontrer l'emploi particulier *avaler sa langue*.

(c) *Le grec était une **langue** morte bien pendue.*
Jeu de mots dû à la superposition de deux emplois attestés.

(d) *Quelle horreur, il a la **langue** verte !*
Neutralisation de l'emploi *une langue verte* (argotique).

7. Pour le mot *maison* :

(a) * *C'est une glace **maison**.*
Emploi adjectival ambigu en raison de la polysémie sur *glace*.

(b) * *C'est un ingénieur **maison**.*
Emploi subtilement différent du précédent en raison de son application à un animé.

(c) * *La **maison** des Vasa déclinait et risquait de perdre le trône de Suède.*
Polysémie par métonymie.

(d) * *La **maison** mère se situait près du fleuve.*
Superposition d'usages entre l'emploi le plus fréquent et un emploi particulier.

(e) * *Il fait partie de la **maison**.*
Expression idiomatique comportant une métonymie et une isotopie du trait /individu/.

(f) * *La **maison** offre à boire.*
Incohérence sémantique : le verbe attend un individu comme sujet.

(g) *Il a fallu reconstruire toutes ces grandes **maisons** que la guerre avait ruinées.*
Divergence sémantique par oscillation entre deux acceptions.

(h) *C'est un fils de bonne **maison**.*
Superposition de deux emplois.

8. Pour le mot *manger* :

(a) *Il **mange** de la soupe chaude.*
Test "vide" : l'objet est un aliment.

(b) *Lors de l'éclipse la lune va **manger** le soleil.*
Emploi particulier résultant de la personnification de la Lune.

(c) *Ce manteau est **mangé** par les mites.*
Collision entre deux acceptions : *manger* (un aliment) et *manger* (détériorer).

(d) *La jalousie me **mange** le cœur.*
Emploi particulier lorsque le sujet est un sentiment.

(e) *Il a **mangé** des millions.*
L'objet est une somme d'argent.

(f) *Tu veux **manger** un morceau ?*
Expression idiomatique figée.

(g) *On peut apporter son **manger**.*
Utilisation du substantif.

(h) *Cette voiture **mange** trop d'huile.*
Faisceau d'isotopies issues de *voiture* et *huile*.

9. Pour le mot *mère* :

(a) * *Une **mère** indigne n'est plus une **mère**.*
Gestion de la négation entre les deux occurrences, ainsi que de l'adjectif d'évaluation négative sur la première occurrence.

(b) * *La cellule **mère** produit deux cellules filles identiques.*
Emploi adjectival particulier doublé d'une relation mère / fille.

(c) * *Notre **Mère** la Terre est vieille de cinq milliards d'années.*
Gestion de la majuscule et du singulier.

(d) * *C'était la plus gentille **mère** pour tous ces orphelins.*
Contradiction sémantique entre *mère* et *orphelins*.

(e) *L'impératrice Théodora était la **Mère** de la nation.*
Gestion de la majuscule, du singulier et du fait que le complément d'attribution est une collection d'individus.

10. Pour le mot *mort* :

(a) *La **mort** vous va si bien.*
Absence d'indice contextuel.

(b) *Il est **mort** d'amour.*
La cause n'est pas un phénomène physique.

(c) *Je m'éclate à **mort**.*
Emploi particulier de *à mort* en tant qu'adverbe.

(d) *J'ai souffert mille amours à **mort**.*
Superposition d'une isotopie issue de *souffrir* et d'un emploi particulier de *à mort* rapporté à un sentiment.

(e) *Il est à deux doigts de la **mort** de froid.*
Superposition de deux emplois correspondant à deux structures syntaxiques attestées.

(f) *Je repose sur mon lit de **mort** éternelle.*
Superposition de deux isotopies.

11. Pour le mot *nez* :

(a) *Cet homme est un sacré **nez**.*
Attribut d'un individu.

(b) *Il fourre son **nez** busqué un peu partout.*
Superposition d'une isotopie issue de *busqué* (qui est un adjectif exclusivement réservé à la qualification de *nez*) et d'un emploi attaché à une structure syntaxique attestée.

(c) *Il lui a rit au **nez**.*
Emploi attesté associé à une structure syntaxique figée.

(d) *Ce chien a du **nez**.*
Structure syntaxique figée dans laquelle le sujet n'est pas une personne.

12. Pour le mot *peau* :

(a) * *Ma **peau** est rêche.*
Absence d'indice contextuel.

(b) * *Je n'aime pas la **peau** du lait.*
Le complément d'attribution est un liquide.

(c) *Je vais lui faire sa vieille **peau** !*
Superposition de deux emplois attestés.

13. Pour le mot *père* :

(a) * *C'est la tradition de nos **pères**.*
Gestion particulière du possessif et du pluriel.

(b) * *Le processus fils est une image du processus **père**.*
Emploi adjectival particulier doublé d'une relation père / fils.

(c) * *Le **père** de Jean n'est pas son vrai **père**.*
Répétition avec une négation et un adjectif sur l'une des occurrences.

(d) * *Les bons **Pères** croyaient apporter la lumière aux contrées sauvages.*
Gestion de la majuscule, du pluriel et de l'adjectif.

(e) *Un **père** en punissant est toujours un **père**.*
Structure tautologique avec une isotopie sur la première occurrence.

(f) *André Breton est le saint **père** du surréalisme.*
Superposition de deux emplois.

14. Pour le mot *roi* :

(a) * *C'est le **roi** de la fête.*
Structure idiomatique : le complément d'attribution est un événement.

(b) * *C'est le **roi** des imbéciles.*
Structure idiomatique : le complément désigne une catégorie définie par un trait caractéristique qui engendre une isotopie.

(c) *Tu es la **reine** des tartes.*
Jeu de mot par dissimilation d'isotopie due à l'homonymie de *tarte*.

(d) * *Le **roi** apparut, entouré de sa cour.*
Emploi attesté par une isotopie.

15. Pour le mot *sœur* :

(a) * *Io possède une lune **sœur**.*
Emploi adjectival sans indice supplémentaire.

(b) *Je cherche l'âme **sœur**.*
Emploi adjectival idiomatique.

(c) *Ô vous, mes **sœurs** d'infortune...*
Gestion du vocatif et d'une isotopie issue d'*infortune*.

16. Pour le mot *soleil* :

(a) * *Les Mayas adoraient le **Soleil**.*
Champ lexical de la religion.

(b) *J'ai fait ma place au **soleil**.*
Expression idiomatique figée.

(c) *Nous avons vu le **soleil** se coucher.*
Emploi attesté par une isotopie.

(d) * *Ma fille est le **soleil** de ma vie.*
Gestion d'une analogie.

(e) * *Les trois jours furent sans **soleil**.*
Isotopie mettant en avant des traits météorologiques.

(f) * *Mon parapluie est couleur **soleil**.*
Emploi adjectival.

(g) * *Certaines planètes gravitent autour de deux **soleils**.*
Champ lexical de l'astronomie.

(h) *Ils sont là, sous la pluie et le **soleil**.*
Faisceau d'isotopies météorologiques.

(i) *L'empire du **Soleil** levant darde ses rayons sur l'occident.*
Superposition d'une expression figée et d'une isotopie météorologique.

(j) *Le cascadeur a effectué une chute **soleil**.*
Emploi adjectival idiomatique.

17. Pour le mot *système* :

(a) *Impossible de supprimer des fichiers **système**.*
Emploi adjectival propre à l'informatique.

(b) *Quel **système** ingénieux !*
Absence d'indice contextuel.

(c) *Reportez-vous au **système** de référence.*
Structure syntaxique attestée dans les domaines des mathématiques et de la physique.

(d) *Windows, ou le **système** oligarchique de la micro-informatique.*
Collision entre le domaine de l'informatique et une structure de syntagme nominal attesté en politique.

18. Pour le mot *toucher* :

(a) *Ne me **touche** pas !*
Gestion de l'impératif.

(b) *Tu es un **touche** à tout...*
Simulation d'un lexème composé : le deuxième élément de la structure *N à N* est générique.

(c) *Ce que tu as dit m'a beaucoup **touché**.*
Glissement du champ conceptuel depuis //sensation// vers //émotion//.

(d) *Ce velours a un **toucher** exquis.*
Le substantif est rapporté à un inanimé.

(e) *Ce commercial a un **toucher** par téléphone très percutant.*
Superposition de deux emplois : substantif rapporté à un individu, et isotopie issue de *téléphone*.

19. Pour le mot *vie* :

(a) * *Il mène une **vie** misérable.*
Emploi idiomatique : *mener une vie*.

(b) * *La **vie** est un long fleuve tranquille.*
Absence d'indice contextuel.

(c) * *Pendant la **vie** d'un projet les gens sont occupés.*
Emploi catachrétique rapporté à un processus.

(d) * *Il perdit la **vie** au cours d un accident.*
Emploi idiomatique attaché à une structure figée.

(e) *On perd sa **vie** à la gagner.*
Collision entre deux emplois attachés à des structures syntaxiques figées.

20. Pour le mot *vieux* :

    (a) *C'est un **vieil** ami.*
    Divergence sémantique par oscillation entre plusieurs interprétations.

    (b) *Ce qu'il est **vieux** jeu !*
    Expression idiomatique.

    (c) *Je l'aime, mon **vieux**.*
    Divergence sémantique par oscillation entre plusieurs interprétations.

21. Pour le mot *voir* :

    (a) *Il **voit** plus loin que le bout de son nez.*
    Expression idiomatique.

    (b) *Il a **vu** son médecin hier.*
    Superposition d'un emploi particulier à l'emploi le plus fréquent.

    (c) *Ca ne se **voit** pas tous les jours un truc pareil.*
    Emploi particulier lié à la pronominalité et au fait que l'objet est générique.

    (d) *Cet homme, **voyez** comme il souffre.*
    Gestion de l'impératif.

    (e) *Je **vois** venir ce mariage de loin et d'un mauvais œil.*
    Superposition de deux emplois liés à des structures idiomatiques.

    (f) *Tu t'es **vu** quand t'as bu !*
    Gestion de la pronominalité quand l'objet est un individu.

A.3 Résultats des jeux d'essais

Nous donnons ci-dessous, pour chaque phrase du corpus de test, le contenu du fichier de résultats engendré par le programme de test. Chaque fichier est composé de deux sections :

1. la liste des propositions d'interprétations (i.e. les vues conceptuelles saillantes ou niées) accompagnées de l'estimation numérique de leur participation au sens de l'occurence, et de leur contexte sous forme d'une paire *[domaine, notion]* ;

2. le détail du potentiel sémantique de l'entrée lexicale considérée.

— *Dans sa **bouche**, les mots prennent un ton plus subtil.*

```
Propositions d'interprétation :
-------------------------------
  cavité_buccale : 0.666667
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
```

```
      cavité_buccale --> Salient
        contenu --> Salient
        appétit --> Valid
        parole --> Salient
      lèvres --> Valid
    personne --> Valid
      convive --> Valid
      personne_à_charge --> Valid
      gourmet --> Valid
      personne_difficile --> Valid
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *Les pompiers font la fine* ***bouche****.*

```
Propositions d'interprétation :
-------------------------------
  personne_difficile : 1.000000
    (domaine = quotidien, notion = personne)
  bouche_d_incendie : 0.500000
    (domaine = technique, notion = ustensile)
  cavité_buccale : 0.000000
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
```

```
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Valid
        parole --> Valid
      lèvres --> Valid
    personne --> Salient
      convive --> Valid
      personne_à_charge --> Valid
      gourmet --> Valid
      personne_difficile --> Salient
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Salient
      bouche_d_incendie --> Salient
        pompiers --> Salient
        eau --> Salient
        pression --> Valid
        incendie --> Valid
```

— *Elle porte un doigt à sa **bouche**.*

```
Propositions d'interprétation :
-------------------------------
  cavité_buccale : 0.000000
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Valid
        parole --> Valid
```

```
      lèvres --> Valid
    personne --> Valid
      convive --> Valid
      personne_à_charge --> Valid
      gourmet --> Valid
      personne_difficile --> Valid
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *La famine chassait les hommes de la montagne, effrayés par le poids des* ***bouches*** *à nourrir.*

```
Propositions d'interprétation :
-------------------------------
  convive : 1.000000
    (domaine = quotidien, notion = personne)
  personne_à_charge : 1.000000
    (domaine = quotidien, notion = personne)
  cavité_buccale : 0.000000
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Valid
        parole --> Valid
```

```
      lèvres --> Valid
    personne --> Salient
      convive --> Salient
      personne_à_charge --> Salient
      gourmet --> Valid
      personne_difficile --> Valid
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *En psychanalyse, Gallimard publie la* ***Bouche*** *de l'inconscient.*

```
Propositions d'interprétation :
-------------------------------
  ouverture : 0.333333
    (domaine = technique, notion = trou)
  cavité_buccale : 0.000000
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Valid
        parole --> Valid
      lèvres --> Valid
    personne --> Valid
```

```
      convive --> Valid
      personne_à_charge --> Valid
      gourmet --> Valid
      personne_difficile --> Valid
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Salient
      ouverture --> Salient
        four --> Valid
        enceinte --> Valid
        profondeur --> Salient
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *Les textes de Brillat-Savarin font venir l'eau aux fines* ***bouches****.*

```
Propositions d'interprétation :
-------------------------------
  gourmet : 1.000000
    (domaine = quotidien, notion = personne)
  personne_difficile : 1.000000
    (domaine = quotidien, notion = personne)
  cavité_buccale : 0.333333
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Salient
        parole --> Valid
      lèvres --> Valid
    personne --> Salient
```

```
      convive --> Valid
      personne_à_charge --> Valid
      gourmet --> Salient
      personne_difficile --> Salient
    moment --> Valid
      fin --> Valid
  militaire --> Valid
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Valid
      orifice --> Valid
        arme_à_feu --> Valid
        projectile --> Valid
        détonation --> Valid
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *Clyde déposa un baiser sur la* ***bouche*** *tiède de son colt.*

```

Propositions d'interprétation :
-------------------------------
  orifice : 1.000000
    (domaine = militaire, notion = composant)
  cavité_buccale : 0.000000
    (domaine = quotidien, notion = anatomie)

Potentiel sémantique :
----------------------
bouche =
  quotidien --> Salient
    anatomie --> Salient
      cavité_buccale --> Salient
        contenu --> Valid
        appétit --> Valid
        parole --> Valid
      lèvres --> Valid
    personne --> Valid
      convive --> Valid
      personne_à_charge --> Valid
```

```
      gourmet --> Valid
      personne_difficile --> Valid
    moment --> Valid
      fin --> Valid
  militaire --> Salient
    arme --> Valid
      canon --> Valid
        explosion --> Valid
        bruit --> Valid
        chaleur --> Valid
    composant --> Salient
      orifice --> Salient
        arme_à_feu --> Salient
        projectile --> Salient
        détonation --> Salient
  géographique --> Valid
    extrémité --> Valid
      embouchure --> Valid
        fleuve --> Valid
        mer --> Valid
      entrée --> Valid
        golfe --> Valid
        détroit --> Valid
        mer --> Valid
  technique --> Valid
    trou --> Valid
      ouverture --> Valid
        four --> Valid
        enceinte --> Valid
        profondeur --> Valid
      accès --> Valid
        cave --> Valid
        métro --> Valid
    ustensile --> Valid
      bouche_d_incendie --> Valid
        pompiers --> Valid
        eau --> Valid
        pression --> Valid
        incendie --> Valid
```

— *C'est une bonne **fille**.*

```

Propositions d'interprétation
-----------------------------
  enfant : 1
    (domaine = quotidien, notion = personne)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
fille =
  quotidien --> Salient
    personne --> Salient
      femme --> Salient
        enfant --> Valid
        jeune --> Valid
      enfant --> Salient
    manière --> Valid
      sans_prévenir --> Valid
        partir --> Valid
    statut --> Valid
      mariée --> Valid
```

```
      mère --> Valid
      servante --> Valid
      religieuse --> Valid
      prostituée --> Valid
    hiérarchie --> Valid
      successeur --> Valid
      généré --> Valid
      similaire --> Valid
    attitude --> Salient
      bienveillance --> Salient
      ingratitude --> Valid
      obéissance --> Valid
      insulte --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
```

— *La cellule mère produit deux cellules* ***filles*** *identiques.*

```
Propositions d'interprétation
-----------------------------
  successeur : 1
    (domaine = quotidien, notion = hiérarchie)
  généré : 1
    (domaine = quotidien, notion = hiérarchie)
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
fille =
  quotidien --> Salient
    personne --> Valid
      femme --> Valid
        enfant --> Valid
        jeune --> Valid
      enfant --> Valid
    manière --> Valid
      sans_prévenir --> Valid
        partir --> Valid
    statut --> Valid
      mariée --> Valid
      mère --> Valid
      servante --> Valid
      religieuse --> Valid
      prostituée --> Valid
    hiérarchie --> Salient
      successeur --> Salient
      généré --> Salient
      similaire --> Salient
    attitude --> Valid
      bienveillance --> Valid
      ingratitude --> Valid
      obéissance --> Valid
      insulte --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
```

— *C'était une bonne **fille** mais publique.*

```
Propositions d'interprétation
-----------------------------
  enfant : 1
    (domaine = quotidien, notion = personne)
  prostituée : 1
    (domaine = quotidien, notion = statut)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
fille =
  quotidien --> Salient
    personne --> Salient
      femme --> Salient
        enfant --> Valid
        jeune --> Valid
      enfant --> Salient
    manière --> Valid
      sans_prévenir --> Valid
        partir --> Valid
    statut --> Salient
      mariée --> Valid
      mère --> Valid
      servante --> Valid
      religieuse --> Valid
      prostituée --> Salient
    hiérarchie --> Valid
      successeur --> Valid
      généré --> Valid
      similaire --> Valid
    attitude --> Salient
      bienveillance --> Salient
      ingratitude --> Valid
      obéissance --> Valid
      insulte --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
```

— *La **fille** de salle l'était restée.*

```
Propositions d'interprétation
-----------------------------
  enfant : 1
    (domaine = quotidien, notion = personne)
  mariée : 1
    (domaine = quotidien, notion = statut)
  servante : 1
    (domaine = quotidien, notion = statut)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
fille =
  quotidien --> Salient
```

```
    personne --> Salient
      femme --> Salient
        enfant --> Valid
        jeune --> Valid
      enfant --> Salient
    manière --> Valid
      sans_prévenir --> Valid
        partir --> Valid
    statut --> Salient
      mariée --> Negated
      mère --> Valid
      servante --> Salient
      religieuse --> Valid
      prostituée --> Valid
    hiérarchie --> Valid
      successeur --> Valid
      généré --> Valid
      similaire --> Valid
    attitude --> Valid
      bienveillance --> Valid
      ingratitude --> Valid
      obéissance --> Valid
      insulte --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
```

— *Tel père, tel **fils** !*

```
Propositions d'interprétation
-----------------------------
  garçon : 1
    (domaine = quotidien, notion = personne)
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
fils =
  quotidien --> Salient
    personne --> Salient
      garçon --> Salient
    hiérarchie --> Salient
      successeur --> Valid
      généré --> Valid
      similaire --> Salient
    attitude --> Valid
      ingratitude --> Valid
      bienveillance --> Valid
      obéissance --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
      christ --> Valid
        homme --> Valid
```

— *Le processus **fils** est une image du processus père.*

```
Propositions d'interprétation
-----------------------------
  successeur : 1
    (domaine = quotidien, notion = hiérarchie)
  généré : 1
    (domaine = quotidien, notion = hiérarchie)
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
fils =
  quotidien --> Salient
    personne --> Valid
      garçon --> Valid
    hiérarchie --> Salient
      successeur --> Salient
      généré --> Salient
      similaire --> Salient
    attitude --> Valid
      ingratitude --> Valid
      bienveillance --> Valid
      obéissance --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
      christ --> Valid
        homme --> Valid
```

— *C'est un bon **fils** à papa.*

```
Propositions d'interprétation
-----------------------------
  garçon : 1
    (domaine = quotidien, notion = personne)
  successeur : 1
    (domaine = quotidien, notion = hiérarchie)
  ingratitude : 1
    (domaine = quotidien, notion = attitude)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)

Potentiel sémantique
--------------------
fils =
  quotidien --> Salient
    personne --> Salient
      garçon --> Salient
    hiérarchie --> Salient
      successeur --> Salient
      généré --> Valid
      similaire --> Valid
    attitude --> Salient
      ingratitude --> Salient
      bienveillance --> Salient
      obéissance --> Valid
```

```
    titre --> Valid
      insulte --> Valid
  littéraire --> Valid
    descendance --> Valid
      descendant --> Valid
      originaire_de --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
      christ --> Valid
        homme --> Valid
```

— *Il faut préserver l'honneur des* Fils *de France.*

```
Propositions d'interprétation
-----------------------------
  garçon : 1
    (domaine = quotidien, notion = personne)
  descendant : 1
    (domaine = littéraire, notion = descendance)
  originaire_de : 1
    (domaine = littéraire, notion = descendance)

Potentiel sémantique
--------------------
fils =
  quotidien --> Salient
    personne --> Salient
      garçon --> Salient
    hiérarchie --> Valid
      successeur --> Valid
      généré --> Valid
      similaire --> Valid
    attitude --> Valid
      ingratitude --> Valid
      bienveillance --> Valid
      obéissance --> Valid
    titre --> Valid
      insulte --> Valid
  littéraire --> Salient
    descendance --> Salient
      descendant --> Salient
      originaire_de --> Salient
  religion --> Valid
    personne --> Valid
      dieu --> Valid
      christ --> Valid
        homme --> Valid
```

— *Le pire parjure est d'être un faux* ***frère*** *de sang.*

```
Propositions d'interprétation
-----------------------------
  ami : 1
    (domaine = quotidien, notion = sentiment)
  compagnon : 1
    (domaine = quotidien, notion = sentiment)
  hypocryte : 1
    (domaine = quotidien, notion = sentiment)
  solidaire : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
```

```
--------------------
frère =
  quotidien --> Salient
    personne --> Valid
      homme --> Valid
        masculin --> Valid
        famille --> Valid
        fratrie --> Valid
    sentiment --> Salient
      ami --> Negated
      proche --> Valid
      compagnon --> Salient
      hypocryte --> Salient
    hiérarchie --> Salient
      simultané --> Valid
      similaire --> Valid
      solidaire --> Salient
  religion --> Valid
    personne --> Valid
      religieux --> Valid
  littéraire --> Valid
    titre --> Valid
      membre --> Valid
        ordre --> Valid
        association --> Valid
```

— *C'est un **frère** pour moi.*

```
Propositions d'interprétation
-----------------------------
  ami : 1
    (domaine = quotidien, notion = sentiment)
  proche : 1
    (domaine = quotidien, notion = sentiment)
  homme : 0.666667
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
frère =
  quotidien --> Salient
    personne --> Salient
      homme --> Salient
        masculin --> Valid
        famille --> Salient
        fratrie --> Salient
    sentiment --> Salient
      ami --> Salient
      proche --> Salient
      compagnon --> Valid
      hypocryte --> Valid
    hiérarchie --> Valid
      simultané --> Valid
      similaire --> Valid
      solidaire --> Valid
  religion --> Valid
    personne --> Valid
      religieux --> Valid
  littéraire --> Valid
    titre --> Valid
      membre --> Valid
        ordre --> Valid
        association --> Valid
```

— *La France et l'Allemagne sont deux pays **frères**.*

```
Propositions d'interprétation
-----------------------------
  simultané : 1
    (domaine = quotidien, notion = hiérarchie)
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)
  solidaire : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
frère =
  quotidien --> Salient
    personne --> Valid
      homme --> Valid
        masculin --> Valid
        famille --> Valid
        fratrie --> Valid
    sentiment --> Valid
      ami --> Valid
      proche --> Valid
      compagnon --> Valid
      hypocryte --> Valid
    hiérarchie --> Salient
      simultané --> Salient
      similaire --> Salient
      solidaire --> Salient
  religion --> Valid
    personne --> Valid
      religieux --> Valid
  littéraire --> Valid
    titre --> Valid
      membre --> Valid
        ordre --> Valid
        association --> Valid
```

— *Délivrez moi ô mes **frères** !*

```
Propositions d'interprétation
-----------------------------
  ami : 1
    (domaine = quotidien, notion = sentiment)
  proche : 1
    (domaine = quotidien, notion = sentiment)
  religieux : 1
    (domaine = religion, notion = personne)
  homme : 0.666667
    (domaine = quotidien, notion = personne)
  membre : 0
    (domaine = littéraire, notion = titre)

Potentiel sémantique
--------------------
frère =
  quotidien --> Salient
    personne --> Salient
      homme --> Salient
        masculin --> Valid
        famille --> Salient
        fratrie --> Salient
    sentiment --> Salient
      ami --> Salient
```

```
      proche --> Salient
      compagnon --> Valid
      hypocryte --> Valid
    hiérarchie --> Valid
      simultané --> Valid
      similaire --> Valid
      solidaire --> Valid
  religion --> Salient
    personne --> Salient
      religieux --> Salient
  littéraire --> Salient
    titre --> Salient
      membre --> Salient
        ordre --> Valid
        association --> Valid
```

— *Ca se passe à la maison des **jeunes**.*

```
Propositions d'interprétation
-----------------------------
  collectif : 1
    (domaine = quotidien, notion = personne)
  âgé : 1
    (domaine = quotidien, notion = état)
  immature : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
jeune =
  technique --> Valid
    état --> Valid
      immature --> Valid
      fini --> Valid
      insuffisant --> Valid
      récent --> Valid
  quotidien --> Salient
    personne --> Salient
      individu --> Valid
      collectif --> Salient
    animal --> Valid
      individu --> Valid
    état --> Salient
      âgé --> Negated
      immature --> Salient
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
      naïf --> Valid
    relation --> Valid
      cadet --> Valid
```

— *Ce vin est un peu **jeune**.*

```
Propositions d'interprétation
-----------------------------
  immature : 1
    (domaine = technique, notion = état)
  fini : 1
    (domaine = technique, notion = état)
  récent : 1
    (domaine = technique, notion = état)
  âgé : 1
```

```
    (domaine = quotidien, notion = état)
  immature : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
jeune =
  technique --> Salient
    état --> Salient
      immature --> Salient
      fini --> Negated
      insuffisant --> Valid
      récent --> Salient
  quotidien --> Salient
    personne --> Valid
      individu --> Valid
      collectif --> Valid
    animal --> Valid
      individu --> Valid
    état --> Salient
      âgé --> Negated
      immature --> Salient
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
      naïf --> Valid
    relation --> Valid
      cadet --> Valid
```

— *Je me suis senti un peu **jeune** sur ce projet.*

```
Propositions d'interprétation
-----------------------------
  fini : 1
    (domaine = technique, notion = état)
  insuffisant : 1
    (domaine = technique, notion = état)
  âgé : 1
    (domaine = quotidien, notion = état)
  immature : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
jeune =
  technique --> Salient
    état --> Salient
      immature --> Valid
      fini --> Negated
      insuffisant --> Salient
      récent --> Valid
  quotidien --> Salient
    personne --> Valid
      individu --> Valid
      collectif --> Valid
    animal --> Valid
      individu --> Valid
    état --> Salient
      âgé --> Negated
      immature --> Salient
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
      naïf --> Valid
```

```
    relation --> Valid
      cadet --> Valid
```

— *C'est **jeune**, un jeune...*

```
Propositions d'interprétation
-----------------------------
  immature : 1
    (domaine = technique, notion = état)
  fini : 1
    (domaine = technique, notion = état)
  récent : 1
    (domaine = technique, notion = état)
  individu : 1
    (domaine = quotidien, notion = personne)
  âgé : 1
    (domaine = quotidien, notion = état)
  immature : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
jeune =
  technique --> Salient
    état --> Salient
      immature --> Salient
      fini --> Negated
      insuffisant --> Valid
      récent --> Salient
  quotidien --> Salient
    personne --> Salient
      individu --> Salient
      collectif --> Valid
    animal --> Valid
      individu --> Valid
    état --> Salient
      âgé --> Negated
      immature --> Salient
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
      naïf --> Valid
    relation --> Valid
      cadet --> Valid
```

— *C'est jeune, un **jeune**...*

```
Propositions d'interprétation
-----------------------------
  individu : 1
    (domaine = quotidien, notion = personne)
  âgé : 1
    (domaine = quotidien, notion = état)
  immature : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
jeune =
  technique --> Valid
    état --> Valid
      immature --> Valid
      fini --> Valid
```

```
      insuffisant --> Valid
      récent --> Valid
  quotidien --> Salient
    personne --> Salient
      individu --> Salient
      collectif --> Valid
    animal --> Valid
      individu --> Valid
    état --> Salient
      âgé --> Negated
      immature --> Salient
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
      naïf --> Valid
    relation --> Valid
      cadet --> Valid
```

— *Le peuple tirait la **langue**.*

```
Propositions d'interprétation
-----------------------------
  dans_le_besoin : 1
    (domaine = quotidien, notion = état)
  organe : 0
    (domaine = anatomie, notion = organe)

Potentiel sémantique
--------------------
langue =
  linguistique --> Valid
    système --> Valid
      parole --> Valid
        parlée --> Valid
        argotique --> Valid
      symboles --> Valid
  anatomie --> Salient
    organe --> Salient
      organe --> Salient
        déglutition --> Valid
        parole --> Valid
        goût --> Valid
  quotidien --> Salient
    objet --> Valid
      partie --> Valid
        allongée --> Valid
    état --> Salient
      dans_le_besoin --> Salient
    activité --> Valid
      se_moquer --> Valid
    attitude --> Valid
      bavard --> Valid
      discret --> Valid
      silencieux --> Valid
      médisant --> Valid
      rigide --> Valid
  littéraire --> Valid
    activité --> Valid
      pourparlers --> Valid
  cuisine --> Valid
    plat --> Valid
      plat --> Valid
```

— *Il étouffe car il a avalé sa **langue**.*

```
Propositions d'interprétation
-----------------------------
  silencieux : 1
    (domaine = quotidien, notion = attitude)
  organe : 0
    (domaine = anatomie, notion = organe)

Potentiel sémantique
--------------------
langue =
  linguistique --> Valid
    système --> Valid
      parole --> Valid
        parlée --> Valid
        argotique --> Valid
      symboles --> Valid
  anatomie --> Salient
    organe --> Salient
      organe --> Salient
        déglutition --> Valid
        parole --> Valid
        goût --> Valid
  quotidien --> Salient
    objet --> Valid
      partie --> Valid
        allongée --> Valid
    état --> Valid
      dans_le_besoin --> Valid
    activité --> Valid
      se_moquer --> Valid
    attitude --> Salient
      bavard --> Valid
      discret --> Valid
      silencieux --> Salient
      médisant --> Valid
      rigide --> Valid
  littéraire --> Valid
    activité --> Valid
      pourparlers --> Valid
  cuisine --> Valid
    plat --> Valid
      plat --> Valid
```

— *Le grec était une **langue** morte bien pendue.*

```
Propositions d'interprétation
-----------------------------
  bavard : 1
    (domaine = quotidien, notion = attitude)
  parole : 0.5
    (domaine = linguistique, notion = système)
  organe : 0
    (domaine = anatomie, notion = organe)

Potentiel sémantique
--------------------
langue =
  linguistique --> Valid
    système --> Salient
      parole --> Salient
        parlée --> Negated
        argotique --> Valid
      symboles --> Valid
```

```
  anatomie --> Salient
    organe --> Salient
      organe --> Salient
        déglutition --> Valid
        parole --> Valid
        goût --> Valid
  quotidien --> Salient
    objet --> Valid
      partie --> Valid
        allongée --> Valid
    état --> Valid
      dans_le_besoin --> Valid
    activité --> Valid
      se_moquer --> Valid
    attitude --> Salient
      bavard --> Salient
      discret --> Valid
      silencieux --> Valid
      médisant --> Valid
      rigide --> Valid
  littéraire --> Valid
    activité --> Valid
      pourparlers --> Valid
  cuisine --> Valid
    plat --> Valid
      plat --> Valid
```

— *Quelle horreur, il a la **langue** verte !*

```
Propositions d'interprétation
-----------------------------
  parole : 0.5
    (domaine = linguistique, notion = système)
  organe : 0
    (domaine = anatomie, notion = organe)

Potentiel sémantique
--------------------
langue =
  linguistique --> Salient
    système --> Salient
      parole --> Salient
        parlée --> Valid
        argotique --> Salient
      symboles --> Valid
  anatomie --> Salient
    organe --> Salient
      organe --> Salient
        déglutition --> Valid
        parole --> Valid
        goût --> Valid
  quotidien --> Valid
    objet --> Valid
      partie --> Valid
        allongée --> Valid
    état --> Valid
      dans_le_besoin --> Valid
    activité --> Valid
      se_moquer --> Valid
    attitude --> Valid
      bavard --> Valid
      discret --> Valid
      silencieux --> Valid
      médisant --> Valid
      rigide --> Valid
```

```
  littéraire --> Valid
    activité --> Valid
      pourparlers --> Valid
  cuisine --> Valid
    plat --> Valid
      plat --> Valid
```

— *C'est une glace **maison**.*

```
Propositions d'interprétation :
-------------------------------
  entreprise : 1.000000
    (domaine = économie, notion = artisanat)
  artisanal : 1.000000
    (domaine = économie, notion = artisanat)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Salient
      entreprise --> Salient
      artisanal --> Salient
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Valid
    groupe_social --> Valid
      famille --> Valid
        membre --> Valid
        honorable --> Valid
        noblesse --> Valid
      serviteur --> Valid
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *C'est un ingénieur **maison**.*

```
Propositions d'interprétation :
-------------------------------
  interne : 1.000000
    (domaine = économie, notion = statut)
  famille : 0.000000
    (domaine = économie, notion = statut)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Valid
```

```
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Valid
      entreprise --> Valid
      artisanal --> Valid
    statut --> Salient
      interne --> Salient
      famille --> Negated
        domestique --> Valid
  littéraire --> Valid
    groupe_social --> Valid
      famille --> Valid
        membre --> Valid
        honorable --> Valid
        noblesse --> Valid
      serviteur --> Valid
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *La **maison** des Vasa déclinait et risquait de perdre le trône de Suède.*

```
Propositions d'interprétation :
------------------------------
  famille : 0.333333
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Valid
    artisanat --> Valid
      entreprise --> Valid
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Salient
    groupe_social --> Salient
      famille --> Salient
        membre --> Valid
        honorable --> Valid
```

```
        noblesse --> Salient
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *La **maison** mère se situait près du fleuve.*

```
Propositions d'interprétation :
------------------------------
  entreprise : 1.000000
    (domaine = économie, notion = artisanat)
  famille : 0.000000
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Salient
      entreprise --> Salient
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Salient
    groupe_social --> Salient
      famille --> Salient
        membre --> Valid
        honorable --> Valid
        noblesse --> Valid
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *Il fait partie de la **maison**.*

```
Propositions d'interprétation :
------------------------------
  famille : 0.333333
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
----------------------
```

```
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Valid
    artisanat --> Valid
      entreprise --> Valid
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Salient
    groupe_social --> Salient
      famille --> Salient
        membre --> Salient
        honorable --> Valid
        noblesse --> Valid
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *La **maison** offre à boire.*

```
Propositions d'interprétation :
-------------------------------
  entreprise : 1.000000
    (domaine = économie, notion = artisanat)
  famille : 0.000000
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Salient
      entreprise --> Salient
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Salient
```

```
    groupe_social --> Salient
      famille --> Salient
        membre --> Valid
        honorable --> Valid
        noblesse --> Valid
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *Il a fallu reconstruire toutes ces grandes **maisons** que la guerre avait ruinées.*

```
Propositions d'interprétation :
-------------------------------
  bâtiment : 1.000000
    (domaine = quotidien, notion = édifice)
  habitation : 1.000000
    (domaine = quotidien, notion = édifice)
  édifice_spécialisé : 1.000000
    (domaine = quotidien, notion = édifice)
  entreprise : 1.000000
    (domaine = économie, notion = artisanat)
  famille : 0.666667
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
----------------------
maison =
  quotidien --> Salient
    édifice --> Salient
      bâtiment --> Salient
      habitation --> Salient
      édifice_spécialisé --> Negated
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Salient
      entreprise --> Salient
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Valid
    groupe_social --> Salient
      famille --> Salient
        membre --> Valid
        honorable --> Salient
        noblesse --> Salient
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *C'est un fils de bonne **maison**.*

```
Propositions d'interprétation :
------------------------------
  entreprise : 1.000000
    (domaine = économie, notion = artisanat)
  famille : 0.666667
    (domaine = littéraire, notion = groupe_social)
  serviteur : 0.000000
    (domaine = littéraire, notion = groupe_social)

Potentiel sémantique :
---------------------
maison =
  quotidien --> Valid
    édifice --> Valid
      bâtiment --> Valid
      habitation --> Valid
      édifice_spécialisé --> Valid
    logement --> Valid
      foyer --> Valid
  astrologie --> Valid
    zone --> Valid
      division_céleste --> Valid
  économie --> Salient
    artisanat --> Salient
      entreprise --> Salient
      artisanal --> Valid
    statut --> Valid
      interne --> Valid
      famille --> Valid
        domestique --> Valid
  littéraire --> Salient
    groupe_social --> Salient
      famille --> Salient
        membre --> Valid
        honorable --> Salient
        noblesse --> Salient
      serviteur --> Negated
        souverain --> Valid
        civil --> Valid
        militaire --> Valid
        collectif --> Valid
```

— *Il **mange** de la soupe chaude.*

```
Propositions d'interprétation
-----------------------------
  destructif : 1
    (domaine = quotidien, notion = manière)
  aggressif : 1
    (domaine = quotidien, notion = manière)
  se_nourrir : 0
    (domaine = quotidien, notion = absorption)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Salient
      se_nourrir --> Salient
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
```

```
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Salient
      destructif --> Salient
      aggressif --> Salient
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Valid
    absorption --> Valid
      consommer --> Valid
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *Lors de l'éclipse la lune va **manger** le soleil.*

```
Propositions d'interprétation
-----------------------------
  cacher : 1
    (domaine = quotidien, notion = occultation)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Valid
      se_nourrir --> Valid
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Valid
      destructif --> Valid
      aggressif --> Valid
    occultation --> Salient
      oublier --> Valid
      prononcer --> Valid
      cacher --> Salient
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Valid
    absorption --> Valid
```

```
      consommer --> Valid
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *Ce manteau est **mangé** par les mites.*

```
Propositions d'interprétation
-----------------------------
  destructif : 1
    (domaine = quotidien, notion = manière)
  aggressif : 1
    (domaine = quotidien, notion = manière)
  se_nourrir : 0
    (domaine = quotidien, notion = absorption)
  consommer : 0
    (domaine = technique, notion = absorption)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Salient
      se_nourrir --> Salient
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Salient
      destructif --> Salient
      aggressif --> Salient
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Salient
    absorption --> Salient
      consommer --> Salient
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *La jalousie me **mange** le cœur.*

```
Propositions d'interprétation
-----------------------------
```

```
  aggressif : 1
    (domaine = quotidien, notion = manière)
  trouble : 1
    (domaine = quotidien, notion = émotion)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Valid
      se_nourrir --> Valid
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Salient
      destructif --> Valid
      aggressif --> Salient
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Salient
      désir --> Valid
      trouble --> Salient
  technique --> Valid
    absorption --> Valid
      consommer --> Valid
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *Il a **mangé** des millions.*

```

Propositions d'interprétation
-----------------------------
  dépenser : 0
    (domaine = quotidien, notion = destruction)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Valid
      se_nourrir --> Valid
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Salient
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
```

```
      dépenser --> Salient
        vainement --> Valid
        perte --> Valid
    manière --> Valid
      destructif --> Valid
      aggressif --> Valid
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Valid
    absorption --> Valid
      consommer --> Valid
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *Tu veux **manger** un morceau ?*

```
Propositions d'interprétation
-----------------------------
  destructif : 1
    (domaine = quotidien, notion = manière)
  aggressif : 1
    (domaine = quotidien, notion = manière)
  se_nourrir : 0
    (domaine = quotidien, notion = absorption)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Salient
      se_nourrir --> Salient
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Salient
      destructif --> Salient
      aggressif --> Salient
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Valid
    absorption --> Valid
      consommer --> Valid
```

```
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *On peut apporter son **manger**.*

```
Propositions d'interprétation
-----------------------------
  nourriture : 1
    (domaine = quotidien, notion = absorption)
  repas : 1
    (domaine = quotidien, notion = absorption)
  appétit : 1
    (domaine = quotidien, notion = absorption)
  se_nourrir : 0
    (domaine = quotidien, notion = absorption)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Salient
      se_nourrir --> Salient
        aliment --> Valid
      nourriture --> Salient
      repas --> Salient
      appétit --> Salient
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Valid
      destructif --> Valid
      aggressif --> Valid
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Valid
    absorption --> Valid
      consommer --> Valid
        carburant --> Valid
        fonctionnement --> Valid
    destruction --> Valid
      entamer --> Valid
      ronger --> Valid
      abîmer --> Valid
```

— *Cette voiture **mange** trop d'huile.*

```
Propositions d'interprétation
-----------------------------
  destructif : 1
```

```
    (domaine = quotidien, notion = manière)
  aggressif : 1
    (domaine = quotidien, notion = manière)
  ronger : 1
    (domaine = technique, notion = destruction)
  consommer : 0.5
    (domaine = technique, notion = absorption)

Potentiel sémantique
--------------------
manger =
  quotidien --> Salient
    absorption --> Valid
      se_nourrir --> Valid
        aliment --> Valid
      nourriture --> Valid
      repas --> Valid
      appétit --> Valid
    destruction --> Valid
      révéler --> Valid
        secret --> Valid
        aveux --> Valid
        délation --> Valid
      dépenser --> Valid
        vainement --> Valid
        perte --> Valid
    manière --> Salient
      destructif --> Salient
      aggressif --> Salient
    occultation --> Valid
      oublier --> Valid
      prononcer --> Valid
      cacher --> Valid
    émotion --> Valid
      désir --> Valid
      trouble --> Valid
  technique --> Salient
    absorption --> Salient
      consommer --> Salient
        carburant --> Salient
        fonctionnement --> Valid
    destruction --> Salient
      entamer --> Valid
      ronger --> Salient
      abîmer --> Valid
```

— *Une **mère** indigne n'est plus une mère.*

```
Propositions d'interprétation
-----------------------------
  générateur : 1
    (domaine = quotidien, notion = hiérarchie)
  génitrice : 1
    (domaine = quotidien, notion = hiérarchie)
  humanité : 1
    (domaine = quotidien, notion = attitude)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mère =
  quotidien --> Salient
```

```
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Salient
      femme --> Salient
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Salient
      générateur --> Salient
      génitrice --> Salient
      antécédant --> Valid
      similaire --> Valid
      principale --> Valid
    attitude --> Salient
      humanité --> Negated
      bienveillance --> Negated
      protection --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Valid
    transcendance --> Valid
      divinité --> Valid
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *Une mère indigne n'est plus une* ***mère****.*

```

Propositions d'interprétation
-----------------------------
  générateur : 1
    (domaine = quotidien, notion = hiérarchie)
  génitrice : 1
    (domaine = quotidien, notion = hiérarchie)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  protection : 1
    (domaine = quotidien, notion = attitude)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mère =
  quotidien --> Salient
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Salient
      femme --> Salient
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Salient
```

```
      générateur --> Salient
      génitrice --> Salient
      antécédant --> Valid
      similaire --> Valid
      principale --> Valid
    attitude --> Salient
      humanité --> Valid
      bienveillance --> Salient
      protection --> Salient
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Valid
    transcendance --> Valid
      divinité --> Valid
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *La cellule* ***mère*** *produit deux cellules filles identiques.*

```
Propositions d'interprétation
-----------------------------
  générateur : 1
    (domaine = quotidien, notion = hiérarchie)
  antécédant : 1
    (domaine = quotidien, notion = hiérarchie)
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)
  principale : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
mère =
  quotidien --> Salient
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Valid
      femme --> Valid
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Salient
      générateur --> Salient
      génitrice --> Valid
      antécédant --> Salient
      similaire --> Salient
      principale --> Salient
    attitude --> Valid
      humanité --> Valid
      bienveillance --> Valid
      protection --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Valid
```

```
    transcendance --> Valid
      divinité --> Valid
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *Notre **Mère** la Terre est vieille de cinq milliards d'années.*

```
Propositions d'interprétation
-----------------------------
  divinité : 1
    (domaine = religion, notion = transcendance)

Potentiel sémantique
--------------------
mère =
  quotidien --> Valid
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Valid
      femme --> Valid
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Valid
      générateur --> Valid
      génitrice --> Valid
      antécédant --> Valid
      similaire --> Valid
      principale --> Valid
    attitude --> Valid
      humanité --> Valid
      bienveillance --> Valid
      protection --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Salient
    transcendance --> Salient
      divinité --> Salient
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *C'était la plus gentille **mère** pour tous ces orphelins.*

```
Propositions d'interprétation
-----------------------------
  générateur : 1
    (domaine = quotidien, notion = hiérarchie)
  génitrice : 1
    (domaine = quotidien, notion = hiérarchie)
  humanité : 1
```

```
    (domaine = quotidien, notion = attitude)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mère =
  quotidien --> Salient
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Salient
      femme --> Salient
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Salient
      générateur --> Negated
      génitrice --> Negated
      antécédant --> Valid
      similaire --> Valid
      principale --> Valid
    attitude --> Salient
      humanité --> Salient
      bienveillance --> Salient
      protection --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Valid
    transcendance --> Valid
      divinité --> Valid
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *L'impératrice Théodora était la **Mère** de la nation.*

```
Propositions d'interprétation
-----------------------------
  générateur : 1
    (domaine = quotidien, notion = hiérarchie)
  génitrice : 1
    (domaine = quotidien, notion = hiérarchie)
  bienveillance : 1
    (domaine = quotidien, notion = attitude)
  protection : 1
    (domaine = quotidien, notion = attitude)
  divinité : 1
    (domaine = religion, notion = transcendance)
  femme : 0
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mère =
  quotidien --> Salient
```

```
    titre --> Valid
      madame --> Valid
        péjoratif --> Valid
        âgé --> Valid
    personne --> Salient
      femme --> Salient
        enfants --> Valid
        famille --> Valid
      partenaire_sexuel --> Valid
        fertile --> Valid
      épouse --> Valid
    hiérarchie --> Salient
      générateur --> Salient
      génitrice --> Salient
      antécédant --> Valid
      similaire --> Valid
      principale --> Valid
    attitude --> Salient
      humanité --> Valid
      bienveillance --> Salient
      protection --> Salient
  littéraire --> Valid
    hiérarchie --> Valid
      cause --> Valid
      origine --> Valid
  religion --> Salient
    transcendance --> Salient
      divinité --> Salient
    personne --> Valid
      religieuse --> Valid
  cuisine --> Valid
    instrument --> Valid
      mère_du_vinaigre --> Valid
```

— *La **mort** vous va si bien.*

```
Propositions d'interprétation
-----------------------------
  phénomènes_vitaux : 1
    (domaine = science, notion = activité)
  défunt : 1
    (domaine = quotidien, notion = personne)
  cadavre : 1
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mort =
  science --> Valid
    activité --> Salient
      phénomènes_vitaux --> Salient
        cessation --> Salient
        ralentissement --> Salient
  littéraire --> Valid
    état --> Valid
      terminal --> Valid
      danger --> Valid
      souffrance --> Valid
      chagrin --> Valid
      épuisement --> Valid
    manière --> Valid
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
```

```
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Valid
    état --> Valid
      damnation --> Valid
  quotidien --> Salient
    processus --> Valid
      arrêt --> Valid
      extinction --> Valid
    manière --> Valid
      mortellement --> Valid
      intensément --> Valid
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Salient
      défunt --> Salient
      cadavre --> Salient
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *Il est **mort** d'amour.*

```
Propositions d'interprétation
-----------------------------
  phénomènes_vitaux : 1
    (domaine = science, notion = activité)
  extinction : 1
    (domaine = quotidien, notion = processus)
  défunt : 1
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
mort =
  science --> Valid
    activité --> Salient
      phénomènes_vitaux --> Salient
        cessation --> Salient
        ralentissement --> Salient
  littéraire --> Valid
    état --> Valid
      terminal --> Valid
      danger --> Valid
      souffrance --> Valid
      chagrin --> Valid
      épuisement --> Valid
    manière --> Valid
```

```
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Valid
    état --> Valid
      damnation --> Valid
  quotidien --> Salient
    processus --> Salient
      arrêt --> Valid
      extinction --> Salient
    manière --> Valid
      mortellement --> Valid
      intensément --> Valid
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Salient
      défunt --> Salient
      cadavre --> Valid
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *Je m'éclate à **mort**.*

```
Propositions d'interprétation
-----------------------------
  mortellement : 1
    (domaine = quotidien, notion = manière)
  intensément : 1
    (domaine = quotidien, notion = manière)

Potentiel sémantique
--------------------
mort =
  science --> Valid
    activité --> Valid
      phénomènes_vitaux --> Valid
        cessation --> Valid
        ralentissement --> Valid
  littéraire --> Valid
    état --> Valid
      terminal --> Valid
      danger --> Valid
      souffrance --> Valid
      chagrin --> Valid
```

```
      épuisement --> Valid
    manière --> Valid
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Valid
    état --> Valid
      damnation --> Valid
  quotidien --> Salient
    processus --> Valid
      arrêt --> Valid
      extinction --> Valid
    manière --> Salient
      mortellement --> Salient
      intensément --> Salient
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Valid
      défunt --> Valid
      cadavre --> Valid
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *J'ai souffert mille amours à **mort**.*

```
Propositions d'interprétation
-----------------------------
  souffrance : 1
    (domaine = littéraire, notion = état)
  mortellement : 1
    (domaine = quotidien, notion = manière)
  intensément : 1
    (domaine = quotidien, notion = manière)

Potentiel sémantique
--------------------
mort =
  science --> Valid
    activité --> Valid
      phénomènes_vitaux --> Valid
        cessation --> Valid
        ralentissement --> Valid
  littéraire --> Salient
    état --> Salient
```

```
      terminal --> Valid
      danger --> Valid
      souffrance --> Salient
      chagrin --> Valid
      épuisement --> Valid
    manière --> Valid
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Valid
    état --> Valid
      damnation --> Valid
  quotidien --> Salient
    processus --> Valid
      arrêt --> Valid
      extinction --> Valid
    manière --> Salient
      mortellement --> Salient
      intensément --> Salient
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Valid
      défunt --> Valid
      cadavre --> Valid
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *Il est à deux doigts de la **mort** de froid.*

```
Propositions d'interprétation
-----------------------------
  phénomènes_vitaux : 1
    (domaine = science, notion = activité)
  terminal : 1
    (domaine = littéraire, notion = état)
  souffrance : 1
    (domaine = littéraire, notion = état)
  extinction : 1
    (domaine = quotidien, notion = processus)

Potentiel sémantique
--------------------
mort =
  science --> Salient
```

```
    activité --> Salient
      phénomènes_vitaux --> Salient
        cessation --> Salient
        ralentissement --> Salient
  littéraire --> Salient
    état --> Salient
      terminal --> Salient
      danger --> Valid
      souffrance --> Salient
      chagrin --> Valid
      épuisement --> Valid
    manière --> Valid
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Valid
    état --> Valid
      damnation --> Valid
  quotidien --> Salient
    processus --> Salient
      arrêt --> Valid
      extinction --> Salient
    manière --> Valid
      mortellement --> Valid
      intensément --> Valid
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Valid
      défunt --> Valid
      cadavre --> Valid
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *Je repose sur mon lit de **mort** éternelle.*

```
Propositions d'interprétation
-----------------------------
  phénomènes_vitaux : 1
    (domaine = science, notion = activité)
  terminal : 1
    (domaine = littéraire, notion = état)
  damnation : 1
    (domaine = religion, notion = état)
```

```
Potentiel sémantique
--------------------
mort =
  science --> Salient
    activité --> Salient
      phénomènes_vitaux --> Salient
        cessation --> Salient
        ralentissement --> Salient
  littéraire --> Salient
    état --> Salient
      terminal --> Salient
      danger --> Valid
      souffrance --> Valid
      chagrin --> Valid
      épuisement --> Valid
    manière --> Valid
      dynamisme --> Valid
      immobilité --> Valid
      peur --> Valid
  droit --> Valid
    peine --> Valid
      peine_capitale --> Valid
      déchéance --> Valid
  psychanalyse --> Valid
    pulsion --> Valid
      autodestruction --> Valid
      morbidité --> Valid
  religion --> Salient
    état --> Salient
      damnation --> Salient
  quotidien --> Valid
    processus --> Valid
      arrêt --> Valid
      extinction --> Valid
    manière --> Valid
      mortellement --> Valid
      intensément --> Valid
      inefficace --> Valid
      hors_d_usage --> Valid
    état --> Valid
      animation --> Valid
      visibilité --> Valid
      mouvement --> Valid
      pratiqué --> Valid
      absence --> Valid
      réaction --> Valid
    personne --> Valid
      défunt --> Valid
      cadavre --> Valid
    place --> Valid
      conducteur --> Valid
        à_côté --> Valid
  jeu --> Valid
    personne --> Valid
      joueur --> Valid
        bridge --> Valid
        quatrième --> Valid
```

— *Cet homme est un sacré **nez**.*

```
Propositions d'interprétation
-----------------------------
  parfumeur : 1
    (domaine = quotidien, notion = personne)
```

```
Potentiel sémantique
--------------------
nez =
  technique --> Valid
    organe --> Valid
      appendice_nasal --> Valid
    extrémité --> Valid
      avant --> Valid
        fuselage --> Valid
      proue --> Valid
      cap --> Valid
      partie --> Valid
        saillante --> Valid
  quotidien --> Salient
    personne --> Salient
      parfumeur --> Salient
    sens --> Valid
      odorat --> Valid
    attitude --> Valid
      perspicace --> Valid
      haine --> Valid
      insolence --> Valid
      ivre --> Valid
      énervement --> Valid
      obéissance --> Valid
    manière --> Valid
      approximativement --> Valid
      en_face --> Valid
      dissimulation --> Valid
    activité --> Valid
      sortir --> Valid
      se_moquer --> Valid
      échapper_à --> Valid
      échouer --> Valid
      s_occuper_de --> Valid
```

— *Il fourre son **nez** busqué un peu partout.*

```
Propositions d'interprétation
-----------------------------
  appendice_nasal : 1
    (domaine = technique, notion = organe)
  s_occuper_de : 1
    (domaine = quotidien, notion = activité)

Potentiel sémantique
--------------------
nez =
  technique --> Salient
    organe --> Salient
      appendice_nasal --> Salient
    extrémité --> Valid
      avant --> Valid
        fuselage --> Valid
      proue --> Valid
      cap --> Valid
      partie --> Valid
        saillante --> Valid
  quotidien --> Salient
    personne --> Valid
      parfumeur --> Valid
    sens --> Valid
      odorat --> Valid
    attitude --> Valid
      perspicace --> Valid
```

```
      haine --> Valid
      insolence --> Valid
      ivre --> Valid
      énervement --> Valid
      obéissance --> Valid
    manière --> Valid
      approximativement --> Valid
      en_face --> Valid
      dissimulation --> Valid
    activité --> Salient
      sortir --> Valid
      se_moquer --> Valid
      échapper_à --> Valid
      échouer --> Valid
      s_occuper_de --> Salient
```

— *Il lui a rit au **nez**.*

```
Propositions d'interprétation
-----------------------------
  appendice_nasal : 1
    (domaine = technique, notion = organe)

Potentiel sémantique
--------------------
nez =
  technique --> Salient
    organe --> Salient
      appendice_nasal --> Salient
    extrémité --> Valid
      avant --> Valid
        fuselage --> Valid
      proue --> Valid
      cap --> Valid
      partie --> Valid
        saillante --> Valid
  quotidien --> Valid
    personne --> Valid
      parfumeur --> Valid
    sens --> Valid
      odorat --> Valid
    attitude --> Valid
      perspicace --> Valid
      haine --> Valid
      insolence --> Valid
      ivre --> Valid
      énervement --> Valid
      obéissance --> Valid
    manière --> Valid
      approximativement --> Valid
      en_face --> Valid
      dissimulation --> Valid
    activité --> Valid
      sortir --> Valid
      se_moquer --> Valid
      échapper_à --> Valid
      échouer --> Valid
      s_occuper_de --> Valid
```

— *Ce chien a du **nez**.*

```
Propositions d'interprétation
-----------------------------
```

```
  odorat : 1
    (domaine = quotidien, notion = sens)
  perspicace : 1
    (domaine = quotidien, notion = attitude)

Potentiel sémantique
--------------------
nez =
  technique --> Valid
    organe --> Valid
      appendice_nasal --> Valid
    extrémité --> Valid
      avant --> Valid
        fuselage --> Valid
      proue --> Valid
      cap --> Valid
      partie --> Valid
        saillante --> Valid
  quotidien --> Salient
    personne --> Valid
      parfumeur --> Valid
    sens --> Salient
      odorat --> Salient
    attitude --> Salient
      perspicace --> Salient
      haine --> Valid
      insolence --> Valid
      ivre --> Valid
      énervement --> Valid
      obéissance --> Valid
    manière --> Valid
      approximativement --> Valid
      en_face --> Valid
      dissimulation --> Valid
    activité --> Valid
      sortir --> Valid
      se_moquer --> Valid
      échapper_à --> Valid
      échouer --> Valid
      s_occuper_de --> Valid
```

— *Ma **peau** est rêche.*

```

Propositions d'interprétation
-----------------------------
  périphérie : 1
    (domaine = technique, notion = surface)
  épiderme : 0
    (domaine = technique, notion = surface)
  croûte : 0
    (domaine = technique, notion = surface)

Potentiel sémantique
--------------------
peau =
  technique --> Salient
    surface --> Salient
      épiderme --> Salient
        organe --> Valid
      cuir --> Valid
      fourrure --> Valid
      enveloppe --> Valid
        végétal --> Valid
      croûte --> Negated
        liquide --> Valid
```

```
        semi_liquide --> Valid
      périphérie --> Salient
    propriété --> Valid
      densité --> Valid
        électricité --> Valid
        croissance --> Valid
        exponentielle --> Valid
  quotidien --> Valid
    état --> Valid
      état_d_esprit --> Valid
      vie --> Valid
    personne --> Valid
      individu --> Valid
        femme --> Valid
    attitude --> Valid
      dureté --> Valid
      à_l_aise --> Valid
      conduite --> Valid
      opinion --> Valid
      amoureux --> Valid
    titre --> Valid
      injure --> Valid
    caractéristique --> Valid
      aspect --> Valid
```

— *Je n'aime pas la **peau** du lait.*

```
Propositions d'interprétation
-----------------------------
  périphérie : 1
    (domaine = technique, notion = surface)
  croûte : 0.5
    (domaine = technique, notion = surface)

Potentiel sémantique
--------------------
peau =
  technique --> Salient
    surface --> Salient
      épiderme --> Valid
        organe --> Valid
      cuir --> Valid
      fourrure --> Valid
      enveloppe --> Valid
        végétal --> Valid
      croûte --> Salient
        liquide --> Salient
        semi_liquide --> Valid
      périphérie --> Salient
    propriété --> Valid
      densité --> Valid
        électricité --> Valid
        croissance --> Valid
        exponentielle --> Valid
  quotidien --> Valid
    état --> Valid
      état_d_esprit --> Valid
      vie --> Valid
    personne --> Valid
      individu --> Valid
        femme --> Valid
    attitude --> Valid
      dureté --> Valid
      à_l_aise --> Valid
      conduite --> Valid
```

```
      opinion --> Valid
      amoureux --> Valid
    titre --> Valid
      injure --> Valid
    caractéristique --> Valid
      aspect --> Valid
```

— *Je vais lui faire sa vieille **peau** !*

```
Propositions d'interprétation
-----------------------------
  vie : 1
    (domaine = quotidien, notion = état)
  individu : 1
    (domaine = quotidien, notion = personne)
  injure : 1
    (domaine = quotidien, notion = titre)

Potentiel sémantique
--------------------
peau =
  technique --> Valid
    surface --> Valid
      épiderme --> Valid
        organe --> Valid
      cuir --> Valid
      fourrure --> Valid
      enveloppe --> Valid
        végétal --> Valid
      croûte --> Valid
        liquide --> Valid
        semi_liquide --> Valid
      périphérie --> Valid
    propriété --> Valid
      densité --> Valid
        électricité --> Valid
        croissance --> Valid
        exponentielle --> Valid
  quotidien --> Salient
    état --> Salient
      état_d_esprit --> Valid
      vie --> Negated
    personne --> Salient
      individu --> Salient
        femme --> Salient
    attitude --> Valid
      dureté --> Valid
      à_l_aise --> Valid
      conduite --> Valid
      opinion --> Valid
      amoureux --> Valid
    titre --> Salient
      injure --> Salient
    caractéristique --> Valid
      aspect --> Valid
```

— *C'est la tradition de nos **pères**.*

```
Propositions d'interprétation :
-------------------------------
  ancêtre : 1.000000
    (domaine = littéraire, notion = hiérarchie)
```

```
Potentiel sémantique :
----------------------
père =
  quotidien --> Valid
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Valid
      géniteur --> Valid
        famille --> Valid
      générateur --> Valid
      antécédant --> Valid
      similaire --> Valid
      responsable --> Valid
        sûr --> Valid
        modeste --> Valid
    attitude --> Valid
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Valid
      éducateur --> Valid
      protecteur --> Valid
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Salient
    hiérarchie --> Salient
      ancêtre --> Salient
      fondateur --> Valid
```

— *Le processus fils est une image du processus **père**.*

```
Propositions d'interprétation :
-------------------------------
  générateur : 1.000000
    (domaine = quotidien, notion = hiérarchie)
  antécédant : 1.000000
    (domaine = quotidien, notion = hiérarchie)
  similaire : 1.000000
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique :
----------------------
père =
  quotidien --> Salient
    titre --> Valid
      monsieur --> Valid
```

```
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Salient
      géniteur --> Valid
        famille --> Valid
      générateur --> Salient
      antécédant --> Salient
      similaire --> Salient
      responsable --> Valid
        sûr --> Valid
        modeste --> Valid
    attitude --> Valid
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Valid
      éducateur --> Valid
      protecteur --> Valid
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      ancêtre --> Valid
      fondateur --> Valid
```

— *Le père de Jean n'est pas son vrai* ***père****.*

```

Propositions d'interprétation :
------------------------------
  générateur : 1.000000
    (domaine = quotidien, notion = hiérarchie)
  géniteur : 0.000000
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique :
----------------------
père =
  quotidien --> Salient
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Salient
      géniteur --> Negated
        famille --> Valid
      générateur --> Negated
      antécédant --> Valid
      similaire --> Valid
      responsable --> Valid
```

```
        sûr --> Valid
        modeste --> Valid
    attitude --> Valid
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Valid
      éducateur --> Valid
      protecteur --> Valid
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      ancêtre --> Valid
      fondateur --> Valid
```

— *Les bons **Pères** croyaient apporter la lumière aux contrées sauvages.*

```
Propositions d'interprétation :
-------------------------------
  prêtre : 1.000000
    (domaine = religion, notion = personne)

Potentiel sémantique :
----------------------
père =
  quotidien --> Valid
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Valid
      géniteur --> Valid
        famille --> Valid
      générateur --> Valid
      antécédant --> Valid
      similaire --> Valid
      responsable --> Valid
        sûr --> Valid
        modeste --> Valid
    attitude --> Valid
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Valid
      éducateur --> Valid
      protecteur --> Valid
  théâtre --> Valid
```

```
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Salient
    personne --> Salient
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Salient
  littéraire --> Valid
    hiérarchie --> Valid
      ancêtre --> Valid
      fondateur --> Valid
```

— *Un **père** en punissant est toujours un père.*

```
Propositions d'interprétation :
-------------------------------
  éducateur : 1.000000
    (domaine = quotidien, notion = attitude)

Potentiel sémantique :
----------------------
père =
  quotidien --> Salient
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Valid
      géniteur --> Valid
        famille --> Valid
      générateur --> Valid
      antécédant --> Valid
      similaire --> Valid
      responsable --> Valid
        sûr --> Valid
        modeste --> Valid
    attitude --> Salient
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Valid
      éducateur --> Salient
      protecteur --> Valid
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
```

```
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
      ancêtre --> Valid
      fondateur --> Valid
```

— *Un père en punissant est toujours un* ***père****.*

```
Propositions d'interprétation :
------------------------------
  bienveillance : 1.000000
    (domaine = quotidien, notion = attitude)
  protecteur : 1.000000
    (domaine = quotidien, notion = attitude)
  responsable : 0.000000
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique :
---------------------
père =
  quotidien --> Salient
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Salient
      géniteur --> Valid
        famille --> Valid
      générateur --> Valid
      antécédant --> Valid
      similaire --> Valid
      responsable --> Salient
        sûr --> Valid
        modeste --> Valid
    attitude --> Salient
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Salient
      éducateur --> Valid
      protecteur --> Salient
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Valid
    personne --> Valid
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Valid
        saint --> Valid
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Valid
    hiérarchie --> Valid
```

```
      ancêtre --> Valid
      fondateur --> Valid
```

— *André Breton est le saint **père** du surréalisme.*

```
Propositions d'interprétation :
-------------------------------
  bienveillance : 1.000000
    (domaine = quotidien, notion = attitude)
  protecteur : 1.000000
    (domaine = quotidien, notion = attitude)
  pape : 1.000000
    (domaine = religion, notion = personne)
  fondateur : 1.000000
    (domaine = littéraire, notion = hiérarchie)
  responsable : 0.000000
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique :
----------------------
père =
  quotidien --> Salient
    titre --> Valid
      monsieur --> Valid
        péjoratif --> Valid
        âge --> Valid
    hiérarchie --> Salient
      géniteur --> Valid
        famille --> Valid
      générateur --> Valid
      antécédant --> Valid
      similaire --> Valid
      responsable --> Salient
        sûr --> Valid
        modeste --> Valid
    attitude --> Salient
      guide --> Valid
        spirituel --> Valid
        conscience --> Valid
        évolution --> Valid
      bienveillance --> Salient
      éducateur --> Valid
      protecteur --> Salient
  théâtre --> Valid
    attitude --> Valid
      grave --> Valid
      digne --> Valid
  religion --> Salient
    personne --> Salient
      dieu --> Valid
        éternel --> Valid
        trinité --> Valid
      pape --> Salient
        saint --> Salient
      théologien --> Valid
        église --> Valid
        antiquité --> Valid
        écrivain --> Valid
      prêtre --> Valid
  littéraire --> Salient
    hiérarchie --> Salient
      ancêtre --> Valid
      fondateur --> Salient
```

— *C'est le **roi** de la fête.*

```
Propositions d'interprétation :
-------------------------------
  principal : 1.000000
    (domaine = quotidien, notion = supériorité)

Potentiel sémantique :
----------------------
roi =
  littéraire --> Valid
    titre --> Valid
      monarque --> Valid
        france --> Valid
        espagne --> Valid
        perse --> Valid
  politique --> Valid
    titre --> Valid
      souverain --> Valid
  quotidien --> Salient
    supériorité --> Salient
      principal --> Salient
      personne --> Valid
        superlatif --> Valid
      lion --> Valid
  jeu --> Valid
    GHOST --> Valid
      pièce --> Valid
        échecs --> Valid
        principal --> Valid
      carte --> Valid
        coeur --> Valid
        carreau --> Valid
        trèfle --> Valid
        pique --> Valid
```

— *C'est le **roi** des imbéciles.*

```
Propositions d'interprétation :
-------------------------------
  personne : 1.000000
    (domaine = quotidien, notion = supériorité)

Potentiel sémantique :
----------------------
roi =
  littéraire --> Valid
    titre --> Valid
      monarque --> Valid
        france --> Valid
        espagne --> Valid
        perse --> Valid
  politique --> Valid
    titre --> Valid
      souverain --> Valid
  quotidien --> Salient
    supériorité --> Salient
      principal --> Valid
      personne --> Salient
        superlatif --> Salient
      lion --> Valid
  jeu --> Valid
    GHOST --> Valid
```

```
      pièce --> Valid
        échecs --> Valid
        principal --> Valid
      carte --> Valid
        coeur --> Valid
        carreau --> Valid
        trèfle --> Valid
        pique --> Valid
```

— *Tu es la **reine** des tartes.*

```
Propositions d'interprétation :
-------------------------------
  principal : 1.000000
    (domaine = quotidien, notion = supériorité)
  personne : 1.000000
    (domaine = quotidien, notion = supériorité)

Potentiel sémantique :
----------------------
roi =
  littéraire --> Valid
    titre --> Valid
      monarque --> Valid
        france --> Valid
        espagne --> Valid
        perse --> Valid
  politique --> Valid
    titre --> Valid
      souverain --> Valid
  quotidien --> Salient
    supériorité --> Salient
      principal --> Salient
      personne --> Salient
        superlatif --> Salient
      lion --> Valid
  jeu --> Valid
    GHOST --> Valid
      pièce --> Valid
        échecs --> Valid
        principal --> Valid
      carte --> Valid
        coeur --> Valid
        carreau --> Valid
        trèfle --> Valid
        pique --> Valid
```

— *Le **roi** apparut, entouré de sa cour.*

```
Propositions d'interprétation
-----------------------------
  souverain : 1
    (domaine = politique, notion = titre)

Potentiel sémantique
--------------------
roi =
  littéraire --> Valid
    titre --> Valid
      monarque --> Valid
        france --> Valid
        espagne --> Valid
```

```
      perse --> Valid
  politique --> Salient
    titre --> Salient
      souverain --> Salient
  quotidien --> Valid
    supériorité --> Valid
      principal --> Valid
      personne --> Valid
        superlatif --> Valid
      lion --> Valid
  jeu --> Valid
    GHOST --> Valid
      pièce --> Valid
        échecs --> Valid
        principal --> Valid
      carte --> Valid
        coeur --> Valid
        carreau --> Valid
        trèfle --> Valid
        pique --> Valid
```

— *Io possède une lune **sœur**.*

```
Propositions d'interprétation
-----------------------------
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)
  simultané : 1
    (domaine = quotidien, notion = hiérarchie)
  apparenté : 1
    (domaine = quotidien, notion = hiérarchie)

Potentiel sémantique
--------------------
soeur =
  religion --> Valid
    personne --> Valid
      religieuse --> Valid
    titre --> Valid
      religieuse --> Valid
  quotidien --> Salient
    personne --> Valid
      femme --> Valid
        famille --> Valid
        fratrie --> Valid
        féminin --> Valid
    hiérarchie --> Salient
      similaire --> Salient
      simultané --> Salient
      apparenté --> Salient
    sentiment --> Valid
      compagnon --> Valid
      optimal --> Valid
      proche --> Valid
```

— *Je cherche l'âme **sœur**.*

```
Propositions d'interprétation
-----------------------------
  similaire : 1
    (domaine = quotidien, notion = hiérarchie)
  simultané : 1
    (domaine = quotidien, notion = hiérarchie)
```

```
  apparenté : 1
    (domaine = quotidien, notion = hiérarchie)
  compagnon : 1
    (domaine = quotidien, notion = sentiment)
  optimal : 1
    (domaine = quotidien, notion = sentiment)

Potentiel sémantique
--------------------
soeur =
  religion --> Valid
    personne --> Valid
      religieuse --> Valid
    titre --> Valid
      religieuse --> Valid
  quotidien --> Salient
    personne --> Valid
      femme --> Valid
        famille --> Valid
        fratrie --> Valid
        féminin --> Valid
    hiérarchie --> Salient
      similaire --> Salient
      simultané --> Salient
      apparenté --> Salient
    sentiment --> Salient
      compagnon --> Salient
      optimal --> Salient
      proche --> Valid
```

— *Ô vous, mes **sœurs** d'infortune...*

```

Propositions d'interprétation
-----------------------------
  religieuse : 1
    (domaine = religion, notion = personne)
  religieuse : 1
    (domaine = religion, notion = titre)
  compagnon : 1
    (domaine = quotidien, notion = sentiment)
  proche : 1
    (domaine = quotidien, notion = sentiment)
  femme : 0.666667
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
soeur =
  religion --> Salient
    personne --> Salient
      religieuse --> Salient
    titre --> Salient
      religieuse --> Salient
  quotidien --> Salient
    personne --> Valid
      femme --> Salient
        famille --> Salient
        fratrie --> Salient
        féminin --> Valid
    hiérarchie --> Valid
      similaire --> Valid
      simultané --> Valid
      apparenté --> Valid
    sentiment --> Salient
      compagnon --> Salient
```

```
      optimal --> Valid
      proche --> Salient
```

— *Les Mayas adoraient le **Soleil**.*

```
Propositions d'interprétation :
------------------------------
  le_soleil : 1.000000
    (domaine = astronomie, notion = objet)
  étoile : 1.000000
    (domaine = astronomie, notion = objet)
  puissance : 1.000000
    (domaine = littéraire, notion = statut)
  gloire : 1.000000
    (domaine = littéraire, notion = statut)
  centre : 1.000000
    (domaine = littéraire, notion = transcendance)
  dieu : 1.000000
    (domaine = littéraire, notion = transcendance)
  source_de_vie : 1.000000
    (domaine = littéraire, notion = transcendance)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Valid
    propriété_physique --> Valid
      lumière --> Valid
      chaleur --> Valid
      beau_temps --> Valid
  astronomie --> Salient
    objet --> Salient
      le_soleil --> Salient
      étoile --> Salient
  littéraire --> Salient
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Salient
      puissance --> Salient
        superlatif --> Salient
      gloire --> Salient
      dirigeant --> Valid
    transcendance --> Salient
      centre --> Salient
      dieu --> Salient
      source_de_vie --> Salient
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *J'ai fait ma place au **soleil**.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Salient
      beau_temps --> Valid
  astronomie --> Valid
    objet --> Valid
      le_soleil --> Valid
      étoile --> Valid
  littéraire --> Valid
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *Nous avons vu le **soleil** se coucher.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  le_soleil : 1.000000
    (domaine = astronomie, notion = objet)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
```

```
      lumière --> Salient
      chaleur --> Salient
      beau_temps --> Valid
  astronomie --> Salient
    objet --> Salient
      le_soleil --> Salient
      étoile --> Valid
  littéraire --> Valid
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *Ma fille est le **soleil** de ma vie.*

```
Propositions d'interprétation :
-------------------------------
  puissance : 1.000000
    (domaine = littéraire, notion = statut)
  gloire : 1.000000
    (domaine = littéraire, notion = statut)
  centre : 1.000000
    (domaine = littéraire, notion = transcendance)
  source_de_vie : 1.000000
    (domaine = littéraire, notion = transcendance)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Valid
    propriété_physique --> Valid
      lumière --> Valid
      chaleur --> Valid
      beau_temps --> Valid
  astronomie --> Valid
    objet --> Valid
      le_soleil --> Valid
      étoile --> Valid
  littéraire --> Salient
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
```

```
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Salient
      puissance --> Salient
        superlatif --> Salient
      gloire --> Salient
      dirigeant --> Valid
    transcendance --> Salient
      centre --> Salient
      dieu --> Valid
      source_de_vie --> Salient
  politique --> Valid
    nation --> Valid
      japon --> Valid
  botanique --> Valid
    espèce --> Valid
      tournesol --> Valid
  sport --> Valid
    mouvement --> Valid
      tour_complet --> Valid
```

— *Les trois jours furent sans **soleil**.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  beau_temps : 1.000000
    (domaine = quotidien, notion = propriété_physique)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Negated
      chaleur --> Negated
      beau_temps --> Negated
  astronomie --> Valid
    objet --> Valid
      le_soleil --> Valid
      étoile --> Valid
  littéraire --> Valid
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
```

```
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *Mon parapluie est couleur **soleil**.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  beau_temps : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  jaune : 0.000000
    (domaine = littéraire, notion = couleur)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Valid
      beau_temps --> Salient
  astronomie --> Valid
    objet --> Valid
      le_soleil --> Valid
      étoile --> Valid
  littéraire --> Salient
    lieu --> Valid
      orient --> Valid
    couleur --> Salient
      jaune --> Salient
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *Certaines planètes gravitent autour de deux **soleils**.*

```
Propositions d'interprétation :
-------------------------------
```

```
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  étoile : 1.000000
    (domaine = astronomie, notion = objet)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Salient
      beau_temps --> Valid
  astronomie --> Salient
    objet --> Salient
      le_soleil --> Valid
      étoile --> Salient
  littéraire --> Valid
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *Ils sont là, sous la pluie et le* ***soleil****.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  beau_temps : 1.000000
    (domaine = quotidien, notion = propriété_physique)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Salient
```

```
      beau_temps --> Salient
  astronomie --> Valid
    objet --> Valid
      le_soleil --> Valid
      étoile --> Valid
  littéraire --> Valid
    lieu --> Valid
      orient --> Valid
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Inhibited
    nation --> Inhibited
      japon --> Inhibited
  botanique --> Inhibited
    espèce --> Inhibited
      tournesol --> Inhibited
  sport --> Inhibited
    mouvement --> Inhibited
      tour_complet --> Inhibited
```

— *L'empire du **Soleil** levant darde ses rayons sur l'occident.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  chaleur : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  beau_temps : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  le_soleil : 1.000000
    (domaine = astronomie, notion = objet)
  orient : 1.000000
    (domaine = littéraire, notion = lieu)
  japon : 1.000000
    (domaine = politique, notion = nation)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Salient
      beau_temps --> Salient
  astronomie --> Salient
    objet --> Salient
      le_soleil --> Salient
      étoile --> Valid
  littéraire --> Salient
    lieu --> Salient
      orient --> Salient
```

```
    couleur --> Valid
      jaune --> Valid
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
  politique --> Salient
    nation --> Salient
      japon --> Salient
  botanique --> Valid
    espèce --> Valid
      tournesol --> Inhibited
  sport --> Valid
    mouvement --> Valid
      tour_complet --> Inhibited
```

— *Le cascadeur a effectué une chute **soleil**.*

```
Propositions d'interprétation :
-------------------------------
  lumière : 1.000000
    (domaine = quotidien, notion = propriété_physique)
  tour_complet : 1.000000
    (domaine = sport, notion = mouvement)
  jaune : 0.000000
    (domaine = littéraire, notion = couleur)

Potentiel sémantique :
----------------------
soleil =
  quotidien --> Salient
    propriété_physique --> Salient
      lumière --> Salient
      chaleur --> Valid
      beau_temps --> Valid
  astronomie --> Inhibited
    objet --> Inhibited
      le_soleil --> Inhibited
      étoile --> Inhibited
  littéraire --> Salient
    lieu --> Valid
      orient --> Valid
    couleur --> Salient
      jaune --> Salient
        gai --> Valid
        lumineux --> Valid
        brûlant --> Valid
    statut --> Valid
      puissance --> Valid
        superlatif --> Valid
      gloire --> Valid
      dirigeant --> Valid
    transcendance --> Valid
      centre --> Valid
      dieu --> Valid
      source_de_vie --> Valid
```

```
  politique --> Valid
    nation --> Valid
      japon --> Inhibited
  botanique --> Valid
    espèce --> Valid
      tournesol --> Inhibited
  sport --> Salient
    mouvement --> Salient
      tour_complet --> Salient
```

— *Impossible de supprimer des fichiers* ***système****.*

```
Propositions d'interprétation
-----------------------------
  système_d_exploitation : 1
    (domaine = informatique, notion = processus)

Potentiel sémantique
--------------------
système =
  littéraire --> Valid
    ensemble --> Valid
      ensemble_d_idées --> Valid
        scientifique --> Valid
        philosophique --> Valid
  informatique --> Salient
    processus --> Salient
      système_d_exploitation --> Salient
        appartenant_à --> Salient
      logiciels --> Valid
    combinaison --> Valid
      architecture_logicielle --> Valid
        distribué --> Valid
      logiciels --> Valid
    traitement --> Valid
      système_expert --> Valid
        intelligence_artificielle --> Valid
  psychanalyse --> Valid
    combinaison --> Valid
      instance --> Valid
  finance --> Valid
    combinaison --> Valid
      système_monétaire --> Valid
  quotidien --> Valid
    fonction --> Valid
      méthodes --> Valid
    finalité --> Valid
      moyen --> Valid
        habile --> Valid
    combinaison --> Valid
      équilibre_émotionnel --> Valid
    artefact --> Valid
      dispositif --> Valid
        fonction_déterminée --> Valid
  politique --> Valid
    organisation --> Valid
      gouvernement --> Valid
      administration --> Valid
      fonctionnement --> Valid
  science --> Valid
    combinaison --> Valid
      ensemble_d_unités --> Valid
        fonctionnement_global --> Valid
      organes --> Valid
        même_nature --> Valid
```

```
  linguistique --> Valid
    combinaison --> Valid
      unités --> Valid
        définitions_réciproques --> Valid
        minimal --> Valid
        phonologique --> Valid
        sémantique --> Valid
  maths --> Valid
    combinaison --> Valid
      critères --> Valid
        équations --> Valid
        contraintes --> Valid
  physique --> Valid
    combinaison --> Valid
      référentiel --> Valid
        cinétique --> Valid
        point_matériel --> Valid
    définition --> Valid
      système_international --> Valid
        unités --> Valid
```

— *Quel **système** ingénieux !*

```
Propositions d'interprétation
-----------------------------
  méthodes : 1
    (domaine = quotidien, notion = fonction)
  moyen : 0
    (domaine = quotidien, notion = finalité)
  dispositif : 0
    (domaine = quotidien, notion = artefact)

Potentiel sémantique
--------------------
système =
  littéraire --> Valid
    ensemble --> Valid
      ensemble_d_idées --> Valid
        scientifique --> Valid
        philosophique --> Valid
  informatique --> Valid
    processus --> Valid
      système_d_exploitation --> Valid
        appartenant_à --> Valid
      logiciels --> Valid
    combinaison --> Valid
      architecture_logicielle --> Valid
        distribué --> Valid
      logiciels --> Valid
    traitement --> Valid
      système_expert --> Valid
        intelligence_artificielle --> Valid
  psychanalyse --> Valid
    combinaison --> Valid
      instance --> Valid
  finance --> Valid
    combinaison --> Valid
      système_monétaire --> Valid
  quotidien --> Salient
    fonction --> Salient
      méthodes --> Salient
    finalité --> Salient
      moyen --> Salient
        habile --> Valid
    combinaison --> Valid
```

```
      équilibre_émotionnel --> Valid
    artefact --> Salient
      dispositif --> Salient
        fonction_déterminée --> Valid
  politique --> Valid
    organisation --> Valid
      gouvernement --> Valid
      administration --> Valid
      fonctionnement --> Valid
  science --> Valid
    combinaison --> Valid
      ensemble_d_unités --> Valid
        fonctionnement_global --> Valid
      organes --> Valid
        même_nature --> Valid
  linguistique --> Valid
    combinaison --> Valid
      unités --> Valid
        définitions_réciproques --> Valid
        minimal --> Valid
        phonologique --> Valid
        sémantique --> Valid
  maths --> Valid
    combinaison --> Valid
      critères --> Valid
        équations --> Valid
        contraintes --> Valid
  physique --> Valid
    combinaison --> Valid
      référentiel --> Valid
        cinétique --> Valid
        point_matériel --> Valid
    définition --> Valid
      système_international --> Valid
        unités --> Valid
```

— *Reportez-vous au **système** de référence.*

```
Propositions d'interprétation
-----------------------------
  critères : 0.5
    (domaine = maths, notion = combinaison)
  référentiel : 0
    (domaine = physique, notion = combinaison)

Potentiel sémantique
--------------------
système =
  littéraire --> Valid
    ensemble --> Valid
      ensemble_d_idées --> Valid
        scientifique --> Valid
        philosophique --> Valid
  informatique --> Valid
    processus --> Valid
      système_d_exploitation --> Valid
        appartenant_à --> Valid
      logiciels --> Valid
    combinaison --> Valid
      architecture_logicielle --> Valid
        distribué --> Valid
      logiciels --> Valid
    traitement --> Valid
      système_expert --> Valid
        intelligence_artificielle --> Valid
```

```
  psychanalyse --> Valid
    combinaison --> Valid
      instance --> Valid
  finance --> Valid
    combinaison --> Valid
      système_monétaire --> Valid
  quotidien --> Valid
    fonction --> Valid
      méthodes --> Valid
    finalité --> Valid
      moyen --> Valid
        habile --> Valid
    combinaison --> Valid
      équilibre_émotionnel --> Valid
    artefact --> Valid
      dispositif --> Valid
        fonction_déterminée --> Valid
  politique --> Valid
    organisation --> Valid
      gouvernement --> Valid
      administration --> Valid
      fonctionnement --> Valid
  science --> Valid
    combinaison --> Valid
      ensemble_d_unités --> Valid
        fonctionnement_global --> Valid
      organes --> Valid
        même_nature --> Valid
  linguistique --> Valid
    combinaison --> Valid
      unités --> Valid
        définitions_réciproques --> Valid
        minimal --> Valid
        phonologique --> Valid
        sémantique --> Valid
  maths --> Salient
    combinaison --> Salient
      critères --> Salient
        équations --> Salient
        contraintes --> Valid
  physique --> Salient
    combinaison --> Salient
      référentiel --> Salient
        cinétique --> Valid
        point_matériel --> Valid
    définition --> Valid
      système_international --> Valid
        unités --> Valid
```

— *Windows, ou le **système** oligarchique de la micro-informatique.*

```
Propositions d'interprétation
-----------------------------
  logiciels : 1
    (domaine = informatique, notion = processus)
  logiciels : 1
    (domaine = informatique, notion = combinaison)
  gouvernement : 1
    (domaine = politique, notion = organisation)
  système_d_exploitation : 0
    (domaine = informatique, notion = processus)

Potentiel sémantique
--------------------
système =
```

```
littéraire --> Valid
  ensemble --> Valid
    ensemble_d_idées --> Valid
      scientifique --> Valid
      philosophique --> Valid
informatique --> Salient
  processus --> Salient
    système_d_exploitation --> Salient
      appartenant_à --> Valid
    logiciels --> Salient
  combinaison --> Salient
    architecture_logicielle --> Valid
      distribué --> Valid
    logiciels --> Salient
  traitement --> Valid
    système_expert --> Valid
      intelligence_artificielle --> Valid
psychanalyse --> Valid
  combinaison --> Valid
    instance --> Valid
finance --> Valid
  combinaison --> Valid
    système_monétaire --> Valid
quotidien --> Valid
  fonction --> Valid
    méthodes --> Valid
  finalité --> Valid
    moyen --> Valid
      habile --> Valid
  combinaison --> Valid
    équilibre_émotionnel --> Valid
  artefact --> Valid
    dispositif --> Valid
      fonction_déterminée --> Valid
politique --> Salient
  organisation --> Salient
    gouvernement --> Salient
    administration --> Valid
    fonctionnement --> Valid
science --> Valid
  combinaison --> Valid
    ensemble_d_unités --> Valid
      fonctionnement_global --> Valid
    organes --> Valid
      même_nature --> Valid
linguistique --> Valid
  combinaison --> Valid
    unités --> Valid
      définitions_réciproques --> Valid
      minimal --> Valid
      phonologique --> Valid
      sémantique --> Valid
maths --> Valid
  combinaison --> Valid
    critères --> Valid
      équations --> Valid
      contraintes --> Valid
physique --> Valid
  combinaison --> Valid
    référentiel --> Valid
      cinétique --> Valid
      point_matériel --> Valid
  définition --> Valid
    système_international --> Valid
      unités --> Valid
```

— *Ne me **touche** pas !*

```
Propositions d'interprétation
-----------------------------
  contact : 1
    (domaine = technique, notion = processus)
  contact : 1
    (domaine = technique, notion = état)
  mouvement : 1
    (domaine = quotidien, notion = processus)
  atteindre : 1
    (domaine = quotidien, notion = processus)
  émouvoir : 1
    (domaine = quotidien, notion = processus)

Potentiel sémantique
--------------------
toucher =
  médecine --> Valid
    pratique --> Valid
      examen --> Valid
  musique --> Valid
    propriété --> Valid
      caractère --> Valid
      style --> Valid
  technique --> Salient
    perception --> Valid
      sens --> Valid
      impression --> Valid
    processus --> Salient
      contact --> Salient
    état --> Salient
      contact --> Salient
      contigu --> Valid
  quotidien --> Salient
    processus --> Salient
      mouvement --> Salient
        contact --> Salient
        main --> Salient
      atteindre --> Salient
      concerner --> Valid
      percevoir --> Valid
      aborder --> Valid
      modifier --> Valid
      consommer --> Valid
      émouvoir --> Salient
    expression --> Valid
      communiquer --> Valid
      dire --> Valid
      dissimuler --> Valid
```

— *Tu es un **touche** à tout...*

```
Propositions d'interprétation
-----------------------------
  contact : 1
    (domaine = technique, notion = processus)
  contact : 1
    (domaine = technique, notion = état)
  concerner : 1
    (domaine = quotidien, notion = processus)
  modifier : 1
    (domaine = quotidien, notion = processus)
```

```
Potentiel sémantique
--------------------
toucher =
  médecine --> Valid
    pratique --> Valid
      examen --> Valid
  musique --> Valid
    propriété --> Valid
      caractère --> Valid
      style --> Valid
  technique --> Salient
    perception --> Valid
      sens --> Valid
      impression --> Valid
    processus --> Salient
      contact --> Salient
    état --> Salient
      contact --> Salient
      contigu --> Valid
  quotidien --> Salient
    processus --> Salient
      mouvement --> Valid
        contact --> Salient
        main --> Valid
      atteindre --> Valid
      concerner --> Salient
      percevoir --> Valid
      aborder --> Valid
      modifier --> Salient
      consommer --> Valid
      émouvoir --> Valid
    expression --> Valid
      communiquer --> Valid
      dire --> Valid
      dissimuler --> Valid
```

— *Ce que tu as dit m'a beaucoup **touché**.*

```
Propositions d'interprétation
-----------------------------
  contact : 1
    (domaine = technique, notion = processus)
  contact : 1
    (domaine = technique, notion = état)
  émouvoir : 1
    (domaine = quotidien, notion = processus)

Potentiel sémantique
--------------------
toucher =
  médecine --> Valid
    pratique --> Valid
      examen --> Valid
  musique --> Valid
    propriété --> Valid
      caractère --> Valid
      style --> Valid
  technique --> Salient
    perception --> Valid
      sens --> Valid
      impression --> Valid
    processus --> Salient
      contact --> Salient
    état --> Salient
      contact --> Salient
```

```
      contigu --> Valid
  quotidien --> Salient
    processus --> Salient
      mouvement --> Valid
        contact --> Salient
        main --> Valid
      atteindre --> Valid
      concerner --> Valid
      percevoir --> Valid
      aborder --> Valid
      modifier --> Valid
      consommer --> Valid
      émouvoir --> Salient
    expression --> Valid
      communiquer --> Valid
      dire --> Valid
      dissimuler --> Valid
```

— *Ce velours a un **toucher** exquis.*

```
Propositions d'interprétation
-----------------------------
  sens : 1
    (domaine = technique, notion = perception)
  impression : 1
    (domaine = technique, notion = perception)
  contact : 1
    (domaine = technique, notion = processus)
  contact : 1
    (domaine = technique, notion = état)
  mouvement : 0.5
    (domaine = quotidien, notion = processus)

Potentiel sémantique
--------------------
toucher =
  médecine --> Valid
    pratique --> Valid
      examen --> Valid
  musique --> Valid
    propriété --> Valid
      caractère --> Valid
      style --> Valid
  technique --> Salient
    perception --> Salient
      sens --> Salient
      impression --> Salient
    processus --> Salient
      contact --> Salient
    état --> Salient
      contact --> Salient
      contigu --> Valid
  quotidien --> Valid
    processus --> Salient
      mouvement --> Salient
        contact --> Salient
        main --> Valid
      atteindre --> Valid
      concerner --> Valid
      percevoir --> Valid
      aborder --> Valid
      modifier --> Valid
      consommer --> Valid
      émouvoir --> Valid
    expression --> Valid
```

```
      communiquer --> Valid
      dire --> Valid
      dissimuler --> Valid
```

— *Ce commercial a un **toucher** par téléphone très percutant.*

```
Propositions d'interprétation
-----------------------------
  caractère : 1
    (domaine = musique, notion = propriété)
  style : 1
    (domaine = musique, notion = propriété)
  sens : 1
    (domaine = technique, notion = perception)
  contact : 1
    (domaine = technique, notion = processus)
  contact : 1
    (domaine = technique, notion = état)
  communiquer : 1
    (domaine = quotidien, notion = expression)
  mouvement : 0.5
    (domaine = quotidien, notion = processus)

Potentiel sémantique
--------------------
toucher =
  médecine --> Valid
    pratique --> Valid
      examen --> Valid
  musique --> Salient
    propriété --> Salient
      caractère --> Salient
      style --> Salient
  technique --> Salient
    perception --> Salient
      sens --> Salient
      impression --> Valid
    processus --> Salient
      contact --> Salient
    état --> Salient
      contact --> Salient
      contigu --> Valid
  quotidien --> Salient
    processus --> Valid
      mouvement --> Salient
        contact --> Salient
        main --> Valid
      atteindre --> Valid
      concerner --> Valid
      percevoir --> Valid
      aborder --> Valid
      modifier --> Valid
      consommer --> Valid
      émouvoir --> Valid
    expression --> Salient
      communiquer --> Salient
      dire --> Valid
      dissimuler --> Valid
```

— *Il mène une **vie** misérable.*

```
Propositions d'interprétation :
```

```
------------------------------
  expérience : 1.000000
    (domaine = quotidien, notion = événement)
  organisation : 1.000000
    (domaine = quotidien, notion = événement)
  mode_de_vie : 1.000000
    (domaine = quotidien, notion = manière)

Potentiel sémantique :
---------------------
vie =
  science --> Valid
    activité --> Valid
      phénomènes_vitaux --> Valid
        nutrition --> Valid
        assimilation --> Valid
        croissance --> Valid
        reproduction --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
  religion --> Valid
    état --> Valid
      divinité --> Valid
        bonheur --> Valid
        paradis --> Valid
        mort --> Valid
  littéraire --> Valid
    manière --> Valid
      entrain --> Valid
      dynamisme --> Valid
    oeuvre --> Valid
      biographie --> Valid
    activité --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
        évolution --> Valid
  quotidien --> Salient
    durée --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
      jamais --> Valid
    événement --> Salient
      expérience --> Salient
      organisation --> Salient
      activité --> Valid
      plaisir --> Valid
        excès --> Valid
    manière --> Salient
      nullement --> Valid
      mode_de_vie --> Salient
      insupportable --> Valid
    condition --> Valid
      moyen_matériel --> Valid
        aliment --> Valid
        argent --> Valid
      condition_humaine --> Valid
```

— *La **vie** est un long fleuve tranquille.*

```

Propositions d'interprétation :
------------------------------
```

```
  expérience : 1.000000
    (domaine = quotidien, notion = événement)
  organisation : 1.000000
    (domaine = quotidien, notion = événement)
  existence : 0.000000
    (domaine = quotidien, notion = durée)

Potentiel sémantique :
---------------------
vie =
  science --> Valid
    activité --> Valid
      phénomènes_vitaux --> Valid
        nutrition --> Valid
        assimilation --> Valid
        croissance --> Valid
        reproduction --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
  religion --> Valid
    état --> Valid
      divinité --> Valid
        bonheur --> Valid
        paradis --> Valid
        mort --> Valid
  littéraire --> Valid
    manière --> Valid
      entrain --> Valid
      dynamisme --> Valid
    oeuvre --> Valid
      biographie --> Valid
    activité --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
        évolution --> Valid
  quotidien --> Salient
    durée --> Salient
      existence --> Salient
        naissance --> Valid
        mort --> Valid
      jamais --> Valid
    événement --> Salient
      expérience --> Salient
      organisation --> Salient
      activité --> Valid
      plaisir --> Valid
        excès --> Valid
    manière --> Valid
      nullement --> Valid
      mode_de_vie --> Valid
      insupportable --> Valid
    condition --> Valid
      moyen_matériel --> Valid
        aliment --> Valid
        argent --> Valid
      condition_humaine --> Valid
```

— *Pendant la **vie** d'un projet les gens sont occupés.*

```
Propositions d'interprétation :
-------------------------------
  expérience : 1.000000
```

```
    (domaine = quotidien, notion = événement)
  organisation : 1.000000
    (domaine = quotidien, notion = événement)
  existence : 0.000000
    (domaine = quotidien, notion = durée)

Potentiel sémantique :
---------------------
vie =
  science --> Valid
    activité --> Valid
      phénomènes_vitaux --> Valid
        nutrition --> Valid
        assimilation --> Valid
        croissance --> Valid
        reproduction --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
  religion --> Valid
    état --> Valid
      divinité --> Valid
        bonheur --> Valid
        paradis --> Valid
        mort --> Valid
  littéraire --> Valid
    manière --> Valid
      entrain --> Valid
      dynamisme --> Valid
    oeuvre --> Valid
      biographie --> Valid
    activité --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
        évolution --> Valid
  quotidien --> Salient
    durée --> Salient
      existence --> Salient
        naissance --> Valid
        mort --> Valid
      jamais --> Valid
    événement --> Salient
      expérience --> Salient
      organisation --> Salient
      activité --> Valid
      plaisir --> Valid
        excès --> Valid
    manière --> Valid
      nullement --> Valid
      mode_de_vie --> Valid
      insupportable --> Valid
    condition --> Valid
      moyen_matériel --> Valid
        aliment --> Valid
        argent --> Valid
      condition_humaine --> Valid
```

— *Il perdit la **vie** au cours d un accident.*

```

Propositions d'interprétation :
------------------------------
  existence : 0.500000
    (domaine = science, notion = activité)
```

```
Potentiel sémantique :
----------------------
vie =
  science --> Salient
    activité --> Salient
      phénomènes_vitaux --> Valid
        nutrition --> Valid
        assimilation --> Valid
        croissance --> Valid
        reproduction --> Valid
      existence --> Salient
        naissance --> Valid
        mort --> Salient
  religion --> Valid
    état --> Valid
      divinité --> Valid
        bonheur --> Valid
        paradis --> Valid
        mort --> Valid
  littéraire --> Valid
    manière --> Valid
      entrain --> Valid
      dynamisme --> Valid
    oeuvre --> Valid
      biographie --> Valid
    activité --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
        évolution --> Valid
  quotidien --> Valid
    durée --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
      jamais --> Valid
    événement --> Valid
      expérience --> Valid
      organisation --> Valid
      activité --> Valid
      plaisir --> Valid
        excès --> Valid
    manière --> Valid
      nullement --> Valid
      mode_de_vie --> Valid
      insupportable --> Valid
    condition --> Valid
      moyen_matériel --> Valid
        aliment --> Valid
        argent --> Valid
      condition_humaine --> Valid
```

— *On perd sa **vie** à la gagner.*

```
Propositions d'interprétation :
-------------------------------
  moyen_matériel : 1.000000
    (domaine = quotidien, notion = condition)
  existence : 0.500000
    (domaine = science, notion = activité)

Potentiel sémantique :
----------------------
```

```
vie =
  science --> Salient
    activité --> Salient
      phénomènes_vitaux --> Valid
        nutrition --> Valid
        assimilation --> Valid
        croissance --> Valid
        reproduction --> Valid
      existence --> Salient
        naissance --> Valid
        mort --> Salient
  religion --> Valid
    état --> Valid
      divinité --> Valid
        bonheur --> Valid
        paradis --> Valid
        mort --> Valid
  littéraire --> Valid
    manière --> Valid
      entrain --> Valid
      dynamisme --> Valid
    oeuvre --> Valid
      biographie --> Valid
    activité --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
        évolution --> Valid
  quotidien --> Salient
    durée --> Valid
      existence --> Valid
        naissance --> Valid
        mort --> Valid
      jamais --> Valid
    événement --> Valid
      expérience --> Valid
      organisation --> Valid
      activité --> Valid
      plaisir --> Valid
        excès --> Valid
    manière --> Valid
      nullement --> Valid
      mode_de_vie --> Valid
      insupportable --> Valid
    condition --> Salient
      moyen_matériel --> Salient
        aliment --> Salient
        argent --> Salient
      condition_humaine --> Valid
```

— *C'est un **vieil** ami.*

```
Propositions d'interprétation
-----------------------------
  âgé : 1
    (domaine = technique, notion = état)
  ancien : 1
    (domaine = technique, notion = état)
  longtemps : 1
    (domaine = quotidien, notion = manière)
  vigoureux : 1
    (domaine = quotidien, notion = attitude)
  charmant : 1
    (domaine = quotidien, notion = attitude)
```

```
Potentiel sémantique
--------------------
vieux =
  technique --> Salient
    état --> Salient
      âgé --> Salient
      ancien --> Salient
    processus --> Valid
      vieillir --> Valid
  quotidien --> Salient
    processus --> Valid
      vivre --> Valid
    manière --> Salient
      longtemps --> Salient
      brusquement --> Valid
    état --> Valid
      vieillesse --> Valid
      usé --> Valid
      surrané --> Valid
    personne --> Valid
      individu --> Valid
      parent --> Valid
      ami --> Valid
      vétéran --> Valid
    attitude --> Salient
      vigoureux --> Negated
      charmant --> Negated
```

— *Ce qu'il est **vieux** jeu !*

```

Propositions d'interprétation
-----------------------------
  âgé : 1
    (domaine = technique, notion = état)
  ancien : 1
    (domaine = technique, notion = état)
  usé : 1
    (domaine = quotidien, notion = état)
  surrané : 1
    (domaine = quotidien, notion = état)

Potentiel sémantique
--------------------
vieux =
  technique --> Salient
    état --> Salient
      âgé --> Salient
      ancien --> Salient
    processus --> Valid
      vieillir --> Valid
  quotidien --> Salient
    processus --> Valid
      vivre --> Valid
    manière --> Valid
      longtemps --> Valid
      brusquement --> Valid
    état --> Salient
      vieillesse --> Valid
      usé --> Salient
      surrané --> Salient
    personne --> Valid
      individu --> Valid
      parent --> Valid
      ami --> Valid
      vétéran --> Valid
```

```
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
```

— *Je l'aime, mon **vieux**.*

```
Propositions d'interprétation
-----------------------------
  âgé : 1
    (domaine = technique, notion = état)
  individu : 1
    (domaine = quotidien, notion = personne)
  parent : 1
    (domaine = quotidien, notion = personne)
  ami : 1
    (domaine = quotidien, notion = personne)

Potentiel sémantique
--------------------
vieux =
  technique --> Salient
    état --> Salient
      âgé --> Salient
      ancien --> Valid
    processus --> Valid
      vieillir --> Valid
  quotidien --> Salient
    processus --> Valid
      vivre --> Valid
    manière --> Valid
      longtemps --> Valid
      brusquement --> Valid
    état --> Valid
      vieillesse --> Valid
      usé --> Valid
      surrané --> Valid
    personne --> Salient
      individu --> Salient
      parent --> Salient
      ami --> Negated
      vétéran --> Valid
    attitude --> Valid
      vigoureux --> Valid
      charmant --> Valid
```

— *Il **voit** plus loin que le bout de son nez.*

```
Propositions d'interprétation
-----------------------------
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  concevoir : 1
    (domaine = quotidien, notion = processus)
  prévoir : 1
    (domaine = quotidien, notion = processus)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  perspicace : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)
```

```
Potentiel sémantique
--------------------
voir =
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Valid
    manière --> Valid
      attentivement --> Valid
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
      apparition --> Valid
    processus --> Salient
      assister_à --> Valid
      être_témoin_de --> Valid
      rencontrer --> Valid
      visiter --> Valid
      se_représenter --> Salient
      juger --> Valid
      veiller_à --> Valid
    manière --> Salient
      fréquemment --> Valid
      mentalement --> Salient
  quotidien --> Salient
    processus --> Salient
      montrer --> Valid
      s_éloigner --> Valid
      causer_des_ennuis --> Valid
      apprécier --> Valid
      concevoir --> Salient
      prévoir --> Salient
      essayer --> Valid
      se_produire --> Valid
      attendre --> Valid
    manière --> Salient
      visiblement --> Salient
      péjoratif --> Valid
      perspicace --> Salient
      en_rapport --> Valid
    interjection --> Valid
      rappel_à_l_ordre --> Valid
```

— *Il a **vu** son médecin hier.*

```
Propositions d'interprétation
-----------------------------
  rencontrer : 1
    (domaine = littéraire, notion = processus)
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  fréquemment : 1
    (domaine = littéraire, notion = manière)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)

Potentiel sémantique
--------------------
voir =
```

```
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Valid
    manière --> Valid
      attentivement --> Valid
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
      apparition --> Valid
    processus --> Salient
      assister_à --> Valid
      être_témoin_de --> Valid
      rencontrer --> Salient
      visiter --> Valid
      se_représenter --> Salient
      juger --> Valid
      veiller_à --> Valid
    manière --> Salient
      fréquemment --> Salient
      mentalement --> Salient
  quotidien --> Salient
    processus --> Valid
      montrer --> Valid
      s_éloigner --> Valid
      causer_des_ennuis --> Valid
      apprécier --> Valid
      concevoir --> Valid
      prévoir --> Valid
      essayer --> Valid
      se_produire --> Valid
      attendre --> Valid
    manière --> Salient
      visiblement --> Salient
      péjoratif --> Valid
      perspicace --> Valid
      en_rapport --> Valid
    interjection --> Valid
      rappel_à_l_ordre --> Valid
```

— *Ca ne se **voit** pas tous les jours un truc pareil.*

```
Propositions d'interprétation
-----------------------------
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  se_produire : 1
    (domaine = quotidien, notion = processus)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)

Potentiel sémantique
--------------------
voir =
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Valid
    manière --> Valid
```

```
      attentivement --> Valid
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
      apparition --> Valid
    processus --> Salient
      assister_à --> Valid
      être_témoin_de --> Valid
      rencontrer --> Valid
      visiter --> Valid
      se_représenter --> Salient
      juger --> Valid
      veiller_à --> Valid
    manière --> Salient
      fréquemment --> Valid
      mentalement --> Salient
  quotidien --> Salient
    processus --> Salient
      montrer --> Valid
      s_éloigner --> Valid
      causer_des_ennuis --> Valid
      apprécier --> Valid
      concevoir --> Valid
      prévoir --> Valid
      essayer --> Valid
      se_produire --> Salient
      attendre --> Valid
    manière --> Salient
      visiblement --> Salient
      péjoratif --> Valid
      perspicace --> Valid
      en_rapport --> Valid
    interjection --> Valid
      rappel_à_l_ordre --> Valid
```

— *Cet homme, **voyez** comme il souffre.*

```
Propositions d'interprétation
-----------------------------
  regarder : 1
    (domaine = technique, notion = processus)
  attentivement : 1
    (domaine = technique, notion = manière)
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)

Potentiel sémantique
--------------------
voir =
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Salient
    manière --> Salient
      attentivement --> Salient
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
```

```
      apparition --> Valid
    processus --> Salient
      assister_à --> Valid
      être_témoin_de --> Valid
      rencontrer --> Valid
      visiter --> Valid
      se_représenter --> Salient
      juger --> Valid
      veiller_à --> Valid
    manière --> Salient
      fréquemment --> Valid
      mentalement --> Salient
  quotidien --> Salient
    processus --> Valid
      montrer --> Valid
      s_éloigner --> Valid
      causer_des_ennuis --> Valid
      apprécier --> Valid
      concevoir --> Valid
      prévoir --> Valid
      essayer --> Valid
      se_produire --> Valid
      attendre --> Valid
    manière --> Salient
      visiblement --> Salient
      péjoratif --> Valid
      perspicace --> Valid
      en_rapport --> Valid
    interjection --> Valid
      rappel_à_l_ordre --> Valid
```

— *Je **vois** venir ce mariage de loin et d'un mauvais œil.*

```
Propositions d'interprétation
-----------------------------
  assister_à : 1
    (domaine = littéraire, notion = processus)
  être_témoin_de : 1
    (domaine = littéraire, notion = processus)
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  apprécier : 1
    (domaine = quotidien, notion = processus)
  concevoir : 1
    (domaine = quotidien, notion = processus)
  prévoir : 1
    (domaine = quotidien, notion = processus)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  perspicace : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)

Potentiel sémantique
--------------------
voir =
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Valid
    manière --> Valid
```

```
      attentivement --> Valid
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
      apparition --> Valid
    processus --> Salient
      assister_à --> Salient
      être_témoin_de --> Salient
      rencontrer --> Valid
      visiter --> Valid
      se_représenter --> Salient
      juger --> Valid
      veiller_à --> Valid
    manière --> Salient
      fréquemment --> Valid
      mentalement --> Salient
  quotidien --> Salient
    processus --> Salient
      montrer --> Valid
      s_éloigner --> Valid
      causer_des_ennuis --> Valid
      apprécier --> Negated
      concevoir --> Salient
      prévoir --> Salient
      essayer --> Valid
      se_produire --> Valid
      attendre --> Valid
    manière --> Salient
      visiblement --> Salient
      péjoratif --> Valid
      perspicace --> Salient
      en_rapport --> Valid
    interjection --> Valid
      rappel_à_l_ordre --> Valid
```

— *Tu t'es **vu** quand t'as bu !*

```

Propositions d'interprétation
-----------------------------
  rencontrer : 1
    (domaine = littéraire, notion = processus)
  se_représenter : 1
    (domaine = littéraire, notion = processus)
  fréquemment : 1
    (domaine = littéraire, notion = manière)
  mentalement : 1
    (domaine = littéraire, notion = manière)
  visiblement : 1
    (domaine = quotidien, notion = manière)
  percevoir : 0
    (domaine = technique, notion = processus)

Potentiel sémantique
--------------------
voir =
  technique --> Salient
    processus --> Salient
      percevoir --> Salient
        yeux --> Valid
      regarder --> Valid
    manière --> Valid
      attentivement --> Valid
  littéraire --> Salient
    événement --> Valid
      naissance --> Valid
```

```
    apparition --> Valid
  processus --> Salient
    assister_à --> Valid
    être_témoin_de --> Valid
    rencontrer --> Salient
    visiter --> Valid
    se_représenter --> Salient
    juger --> Valid
    veiller_à --> Valid
  manière --> Salient
    fréquemment --> Salient
    mentalement --> Salient
quotidien --> Salient
  processus --> Valid
    montrer --> Valid
    s_éloigner --> Valid
    causer_des_ennuis --> Valid
    apprécier --> Valid
    concevoir --> Valid
    prévoir --> Valid
    essayer --> Valid
    se_produire --> Valid
    attendre --> Valid
  manière --> Salient
    visiblement --> Salient
    péjoratif --> Valid
    perspicace --> Valid
    en_rapport --> Valid
  interjection --> Valid
    rappel_à_l_ordre --> Valid
```

Annexe B

Introduction à ActiveX

ActiveX est la partie des systèmes d'exploitation WINDOWS 95/NT4 qui est chargée de la mise en œuvre du standard (D)COM. Celui-ci a été élaboré par Microsoft afin de permettre à n'importe quel utilisateur d'étendre le système d'exploitation en ajoutant des modules, ou composants logiciels, additionnels sur sa propre station de travail, ou bien en accédant à des composants situés sur une machine distante, à travers un réseau. ActiveX permet donc de gérer la coopération d'un ensemble de composants logiciels distribués aussi bien entre processus tournant sur une même machine (par exemple, en autorisant plusieurs applications à mettre leurs ressources en commun), qu'entre plusieurs machines participant à un réseau local (Intranet), ou même global (Internet).

ActiveX se fonde sur le standard de conception (D)COM qui tire parti de techniques évoluées du Génie Logiciel, notamment la programmation modulaire, orientée objets par négociation de contrats. Dans la section suivante, nous allons présenter plus précisément ce standard, qui est en fait constitué d'un large ensemble de spécifications. ActiveX est une mise en œuvre particulière de (D)COM et, faisant partie du système d'exploitation, il permet de créer des architectures de composants logiciels distribués et coopératifs sans requérir à une surcouche logicielle.

Les techniques avancées de conception et de programmation qui sont au cœur d'ActiveX ont cependant pour conséquence de rendre plus ardue la réalisation des composants logiciels. L'objet de ce chapitre est de présenter les techniques fondamentales qui ont été utilisées lors de la mise en œuvre du modèle PELEAS, afin de mieux comprendre sa structure et ses interactions possibles avec l'environnement.

B.1 Le standard (D)COM

COM, qui est l'abréviation de *Component Object Model*, est la solution développée par Microsoft afin de résoudre le problème de la coopération et de la réutilisation de nombreux composants logiciels. Trois idées fondamentales président à sa création :

— Un composant logiciel, que ce soit une application indépendante ou juste un module appelé par un autre composant, doit se comporter d'une manière similaire à un objet d'un langage de classes (comme Smalltalk, C++, Eiffel...) Ceci permet d'utiliser un cadre de conception unique pour tous les développeurs.

— Deux composants logiciels doivent pouvoir coopérer même s'ils ont été réalisés avec des langages de programmation différents ou des compilateurs différents. La coopération doit aussi être affran-

chie des différences de représentation en mémoire du code exécutable et des données (on veut donc se libérer des contraintes matérielles concernant les processeurs).

— La distribution d'une nouvelle version d'un composant logiciel ne doit pas perturber sa coopération avec les autres composants. Ceux qui en sont capables peuvent tirer parti de ses nouvelles fonctionnalités, mais les autres doivent pouvoir continuer d'interagir avec lui sans voir de différence entre l'ancienne et la nouvelle version.

On dit de COM qu'il fournit un *standard d'interopérabilité binaire*. C'est-à-dire qu'il permet de concevoir des composants logiciels capables de coopérer grâce à la normalisation de leurs transactions jusqu'au niveau le plus élémentaire d'échange d'informations : le code binaire. Grâce au respect de ce protocole de coopération, les composants interagissent sans se préoccuper des plates-formes matérielles, des supports de transmission des données, ni des langages de programmation utilisés.

Pris tel quel, COM formalise la coopération de composants logiciels résidant sur une même machine, notamment par la spécification des échanges de données et de requêtes entre différents processus qui s'exécutent en parallèle. (D)COM est une extension de COM qui spécifie en outre les échanges entre des composants logiciels installés sur des machines différentes[1]. Ceci est possible grâce à la prise en charge du mécanisme *Distributed Computing Environment Remote Procedure Call* (ou DCE RPC). Ce mécanisme est largement utilisé dans les réseaux hétérogènes pour permettre l'invocation par une machine cliente de routines résidant sur une machine serveur, quels que soient leurs ressources matérielles, leurs systèmes d'exploitation et le protocole réseau utilisé pour les connecter (tout ceci dans un cas idéal, bien entendu).

(D)COM ajoute un certain nombre de spécifications à celles qui composent COM, en particulier la possibilité d'activer un composant logiciel situé sur une machine distante particulière, ainsi que tout ce qui concerne les transactions sécurisées. Cependant, un composant qui a été conçu dans le cadre de COM seul peut également tirer parti des extensions de (D)COM, moyennant une procédure d'installation supplémentaire[2]. En dehors de ce préliminaire, l'utilisation de (D)COM est transparente pour les composants respectant le standard COM. Il n'est donc pas besoin de modifier leur code exécutable. Ceci autorise donc à la fois les développeurs et les utilisateurs à concevoir et utiliser les composants comme s'ils fonctionnaient sur la même machine, de façon concurrente, alors qu'ils sont éventuellement distribués à travers un réseau local, voire même sur Internet.

La figure B.1 montre un cas particulier de coopération entre différents composants logiciels grâce à COM et (D)COM. Sur la machine 1, une application régulière (i.e. qui n'est pas un composant logiciel au sens de COM), interagit simultanément avec un module ActiveX et une application ActiveX. Une application ActiveX est une application autonome capable de fournir des services répondant aux spécifications définies par COM, tandis qu'un module ActiveX n'est pas autonome : on ne peut pas le lancer indépendamment d'une application client. La figure présente le module à l'intérieur de son client car, dans les faits, le système d'exploitation considère que le module ActiveX fonctionne à l'intérieur de l'espace mémoire attribué à l'application (un module ActiveX est en fait une bibliothèque DLL, cf. annexe C).

Ce module, dans l'exemple présenté ici, fait lui-même appel à une application ActiveX située sur une machine distante. La coopération entre ce module et son application serveur se fait par des transactions (D)COM normalisées. Nous avons représenté ici le cas le plus courant, où les deux machines sont connectées par un réseau utilisant le protocole TCP/IP.

[1] A condition que ces machines disposent chacune d'un système d'exploitation supportant ActiveX, bien sûr.

[2] En résumé, il faut informer le système d'exploitation que tel composant peut être trouvé sur telle machine

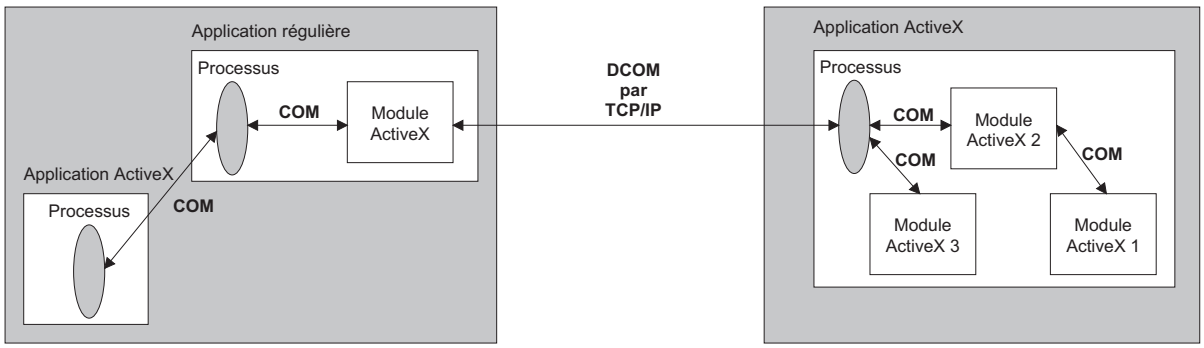

FIG. B.1: Coopération de composants grâce à COM et (D)COM

La coopération entre composants logiciels est, bien entendu, totalement récursive : un composant peut soumettre une requête à un autre composant qui, pour la satisfaire, va soumettre une requête à un troisième composant, qui... C'est ce qu'on peut voir sur la machine 2. L'application ActiveX qui a été invoquée depuis la machine 1 interagit avec deux modules ActiveX. L'un de ces deux modules fait lui-même appel à un troisième module. Tout ceci pourrait se répéter encore de nombreuses fois.

Nous allons maintenant préciser certains termes propres à (D)COM, puis nous montrerons en quoi (D)COM peut être assimilé à un cadre de conception orienté objets. Enfin, nous introduirons la partie d'ActiveX qui se charge de rendre un composant ActiveX *programmable* depuis l'extérieur (au même titre qu'un objet d'un langage orienté objets) : Automation.

B.1.1 Un peu de terminologie

Par la suite, et comme dans la littérature concernée, nous utiliserons indifféremment les termes *objet* COM et *composant logiciel*. La raison de cet abus de langage est fournie dans les paragraphes suivants.

Pour clarifier un peu les choses, notons que COM est une norme, une spécification pour la programmation des objets distribués, tandis qu'ActiveX est la mise en œuvre de COM la plus récente fournie par Microsoft dans ses systèmes d'exploitation de la famille WINDOWS. On pourra aussi entendre parfois parler d'OLE comme mise en œuvre de COM. En fait, à l'origine, OLE était un ensemble de modules du système d'exploitation chargés de partager les documents entre plusieurs applications. Ce sous-système a été mis au point avant que le succès grandissant d'Internet ne conduise Microsoft à intégrer dans ses systèmes d'exploitation des outils de raccordement à Internet, et de distribution des données à travers les très grands réseaux. C'est, entre autres, le rôle que joue l'extension (D)COM. ActiveX est la version d'OLE permettant de travailler avec Internet (plus précisément, ActiveX est OLE 2 avec (D)COM).

Enfin, il faut savoir qu'un objet COM est toujours situé physiquement dans un fichier exécutable du système d'exploitation[1] qui est alors le *serveur de l'objet* COM, ou plus simplement un *serveur* COM. De plus, le même fichier exécutable peut être le serveur de plusieurs objets COM simultanément.

[1]Ce fichier exécutable porte l'extension .EXE s'il s'agit d'une application, .DLL si c'est un module qui doit obligatoirement être appelé par un processus pour s'exécuter, ou .OCX si c'est un *contrôle ActiveX*, c'est-à-dire s'il est spécialement conçu pour être distribué sur Internet.

B.1.2 COM et les interfaces

COM s'inspire très largement de la conception orientée objets, et les spécifications qui le composent sont d'ailleurs très proches du langage C++ qui était, à l'origine, son principal médium de mise en œuvre. La littérature concernée emploie d'ailleurs plutôt le terme *objet* que *composant logiciel*. Nous allons voir que cette dénomination n'est pas entièrement usurpée, même si elle ne correspond pas à une conception classique des objets pour les raisons suivantes : l'utilisation très particulières des interfaces, et l'abandon de la relation d'héritage/composition entre objets au profit de la délégation/agrégation.

Un des buts fondamentaux de COM est de permettre à des composants logiciels d'interagir de façon parallèle, quel que soit le langage de programmation utilisé pour réaliser chacun d'entre eux, et quelles que soient leurs diverses mises en œuvre internes. COM suppose donc une encapsulation stricte de la mise en œuvre de tout objet. La seule façon dont un objet peut communiquer avec l'extérieur est d'exposer un ensemble de *propriétés*, consultables ou pas, et de *méthodes*. Celles-ci sont en fait des points d'entrée permettant à un objet client de soumettre une requête à cet objet, considéré comme serveur.

Nous faisons tout d'abord quelques rappels sur les principes des langages orientés objets à base de classes, tels que C++, Smalltalk, Eiffel, Ceyx...

B.1.2.1 Les classes et leurs relations dans les langages traditionnels

L'activité principale dans ces langages consiste à définir des classes d'objets semblables. Une classe est en fait une spécification, un "moule" servant à produire les objets qui sont des instances de cette classe.

Une classe comporte deux types de spécifications :

1. Des spécifications privées qui décrivent les éléments (données et routines) nécessaires à la mise en œuvre des instances de la classe. Ces éléments relèvent en fait de la "cuisine interne" des instances et c'est pour cette raison qu'elles sont privées, c'est-à-dire non accessibles pour les instances d'une autres classe. C'est ce qu'on appelle le principe d'*encapsulation* : les autres classes n'ont aucune connaissance de ces éléments, principalement afin qu'elles ne puissent pas les modifier inopinément, ce qui risquerait de mettre en danger les contraintes d'intégrité portant sur les données contenues dans l'instance.
2. Des spécifications publiques qui spécifient de quelle manière les instances d'autres classes peuvent interagir avec cette instance. Cela permet notamment de centraliser l'accès aux données de l'instance dans des routines bien précises qui s'assurent que les contraintes d'intégrité des données sont respectées. L'ensemble des spécifications publiques s'appelle l'*interface* de la classe.

On dit de plus qu'un objet est défini par :

— son *état* : l'ensemble des valeurs des données contenues dans l'objet ;

— son *comportement* : l'ensemble des interactions autorisées sur l'objet par l'intermédiaire de l'interface de sa classe ;

— son *identité* : la classe auquel l'objet appartient, plus ce qui fait que deux instances d'une même classe, même si elles sont dans le même état, sont néanmoins deux entités différentes (pour une discussion sur l'importance de la notion d'identité, on peut se reporter au paragraphe traitant de la différence entre un type abstrait et une classe dans Surcin [119, pp. 40-44]).

De manière conventionnelle, on nommera *propriété* d'un objet tout élément de l'interface de sa classe qui permet de consulter ou de modifier un aspect particulier de son état.

De façon converse, on nommera *méthode* un élément de l'interface qui fait réaliser par l'objet une action ne concernant pas uniquement la modification de son état.

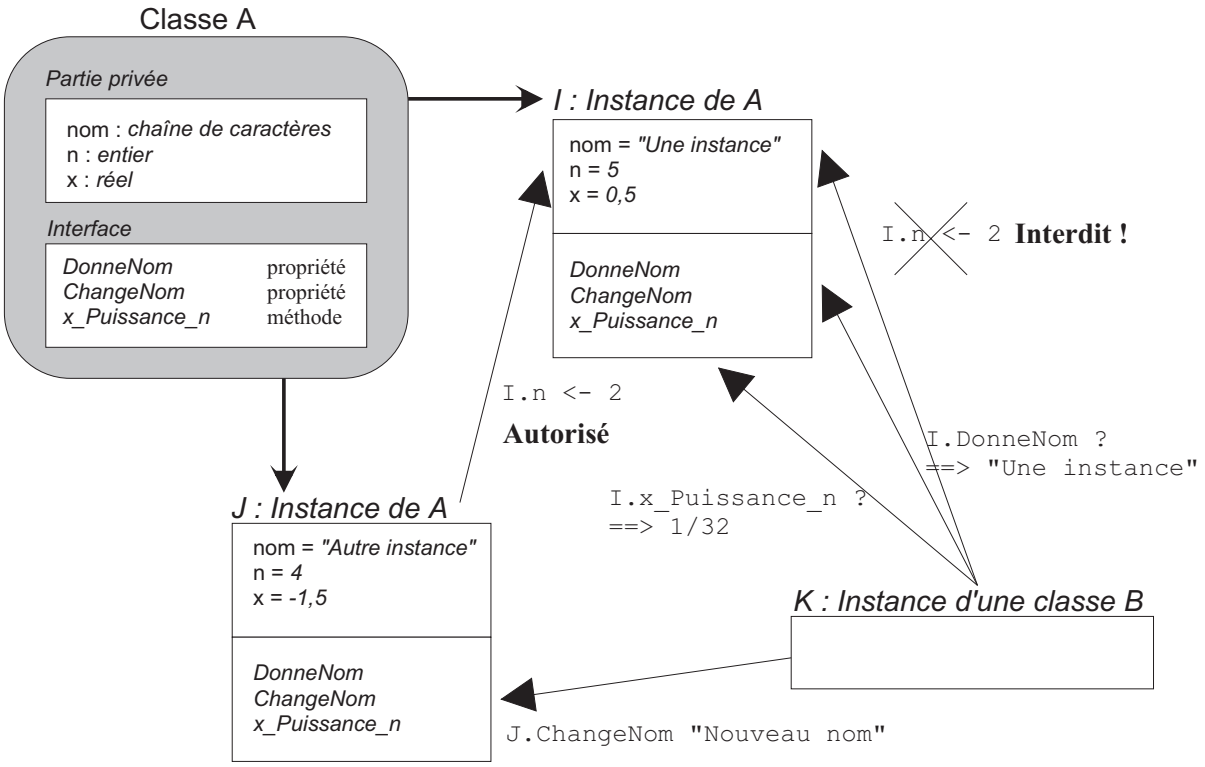

FIG. B.2: Relations entre objets

Tout ceci nous permet de comprendre les relations qu'entretiennent une instance et sa classe ou bien deux instances de classes différentes. Mais les langages orientés objets permettent en outre d'établir des relations directement entre classes, afin de les organiser : la relation de composition et la relation d'héritage.

— la *composition* : une classe A peut utiliser parmi ses éléments privés une instance d'une autre classe B. L'interface de B ne sera pas contenue par l'interface de A. Par conséquent, en plus de la relation de composition, les classes A et B entretiennent une relation de client (A) à serveur (B) qui est le mode naturel de coopération entre instances (cf. figure B.2, où l'instance K de la classe B est client des instances I et J de la classe A) ;

— l'*héritage* : une classe B peut hériter son interface d'une classe A. Elle possède alors *a priori* la même interface que A (et donc le même comportement, voire le même état). Cependant, cette relation d'héritage est généralement utilisée pour *spécialiser* une classe. C'est-à-dire qu'usuellement, les spécifications (autant privées que publiques) de B présenteront des spécificités qui ne figurent pas dans A. Ces spécificités peuvent être des éléments supplémentaires : de nouvelles données privées, de nouvelles propriétés, de nouvelles méthodes. Ou alors il peut s'agir de modifications

apportées à certaines propriétés et/ou méthodes de A : une instance de B présente alors un comportement légèrement différent de celui d'une instance de A. On notera cependant, que la relation d'héritage ne peut pas supprimer d'élément : elle ne fonctionne que par ajout et/ou modification.

La figure B.3 montre une relation d'héritage entre une classe Point et une classe Cercle. La classe Cercle présentée ici hérite des données privées représentant les coordonnées d'un point, car on suppose ici qu'un cercle est défini par son centre et son rayon. Elle hérite également des propriétés permettant d'avoir accès à ces coordonnées.

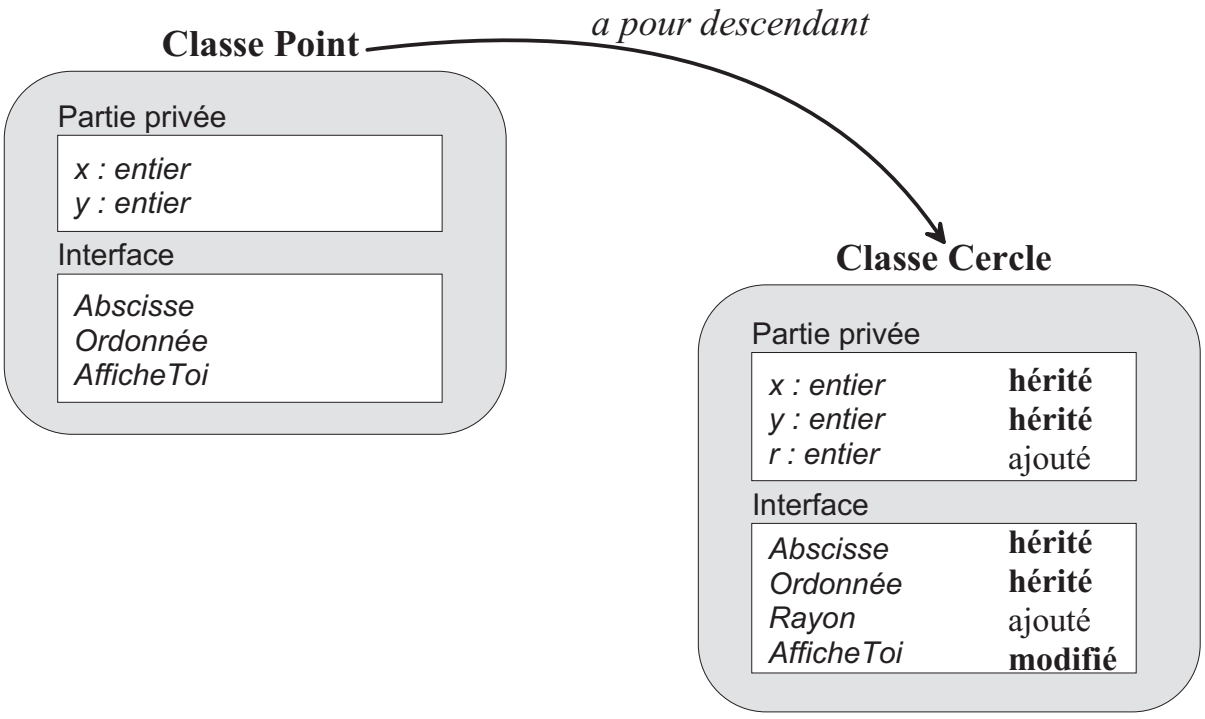

FIG. B.3: Relation d'héritage

La méthode AfficheToi de la classe Cercle est également héritée de la classe Point, mais elle a été modifiée : en effet, même si le point et le cercle partagent le même comportement qui consiste à pouvoir être affiché (à l'écran ici), la mise en œuvre de l'affichage n'est pas la même dans les deux cas.

Enfin, la classe Cercle possède une donnée privée de plus que la classe Point et qui correspond au rayon. On a aussi ajouté une propriété permettant de connaître la valeur du rayon.

B.1.2.2 Les classes de COM

COM utilise également des classes (en fait, COM a été écrit à l'origine pour faciliter sa mise en œuvre en C++), mais celles-ci diffèrent légèrement de ce que nous venons d'exposer. La première différence est plutôt anecdotique puisqu'il s'agit de terminologie : les classes de COM s'appellent des *coclasses* (pour *Component Object Classes*).

Un aspect radicalement nouveau des coclasses par rapport aux classes est qu'une coclasse peut exposer *plusieurs* interfaces. Chaque interface est un ensemble de spécifications pour des méthodes "sémantiquement cohérentes" (selon les critères de leur développeur).

L'activité principale lors du développement avec COM n'est plus la définition des classes, mais la définition des interfaces. Chaque interface est composée d'un ensemble de spécifications de méthodes données sous forme de *prototypes*. Un prototype spécifie le nom de la méthode et le type de valeur qu'elle renvoie, ainsi que les noms et les types de ses paramètres plus leur mode de transmission : en entrée, en sortie, en entrée/sortie. De plus, toute interface doit porter un nom unique et immuable. Une interface spécifie donc une sorte de "contrat" entre le serveur qui la met en œuvre et ses éventuels clients : pour un nom d'interface donné, le client est assuré que le serveur qui met cette interface en œuvre proposera partout et toujours les même services, c'est-à-dire les méthodes de cette interface.

Concevoir un objet COM consiste alors à définir au moins une coclasse pour cet objet en lui donnant un nom et en spécifiant quelles interfaces il doit mettre en œuvre.

ActiveX, étant une mise en œuvre de (D)COM, remplit d'entrée de jeu le système d'exploitation avec de nombreuses interfaces (une centaine environ) et quelques objets les utilisant (l'explorateur Web, un bloc-notes, un calepin pour le dessin, ainsi que d'autres moins visibles mais très utiles pour la gestion de la mémoire et d'autres tâches de bas niveau). Lorsqu'on développe des objets COM, on bute très vite contre le nombre limité d'interfaces disponibles. Il est alors possible de définir de nouvelles interfaces qu'on ajoute au système d'exploitation, et qu'on peut ensuite employer au même titre que les interfaces d'origine.

L'ensemble des interfaces comprises par une coclasse est absolument arbitraire, à ceci près que toute coclasse doit absolument comprendre l'interface `IUnknown`, que nous allons détailler, et toute interface doit hériter (au sens des langages orientés objets) de cette interface.

La figure B.4 schématise très grossièrement les relations entre les spécifications des interfaces, des coclasses, les serveurs qui les mettent en œuvre et les applications qui y recourent.

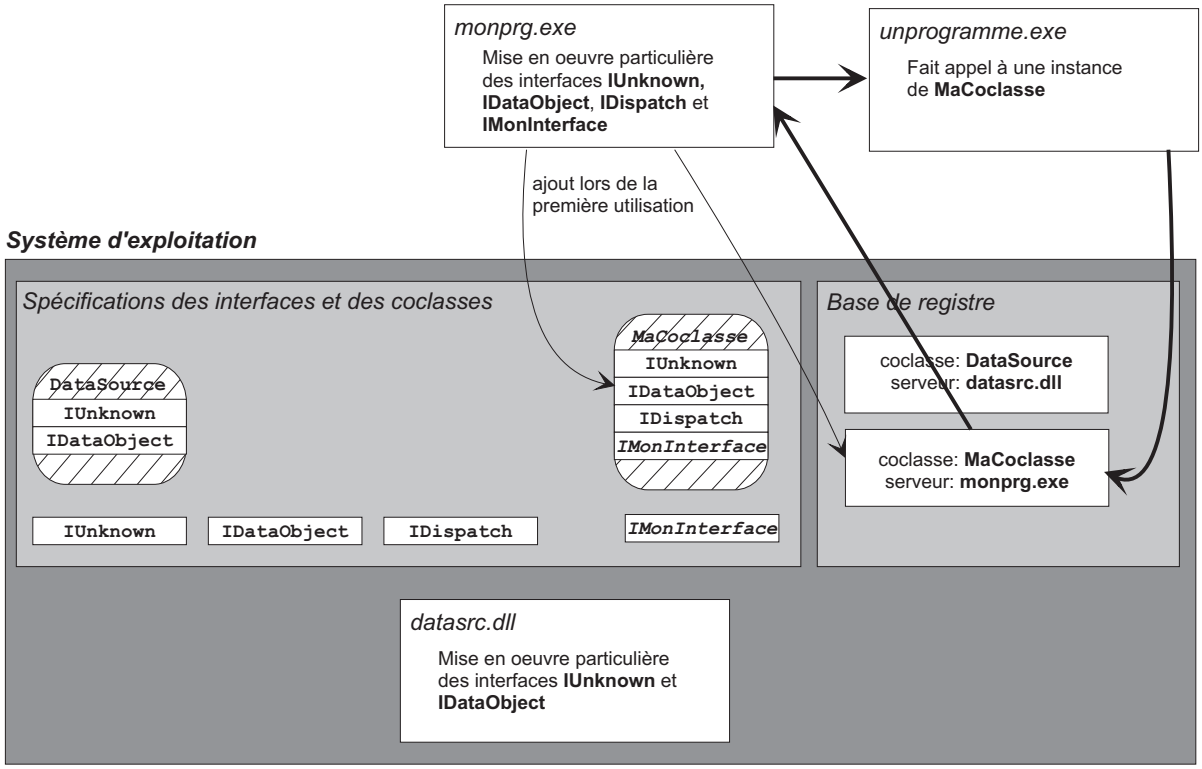

FIG. B.4: Relations entre le système d'exploitation et les composants ActiveX

Comme nous l'avons mentionné plus haut, ActiveX apporte un certain nombre d'interfaces et de coclasses natives dans le système d'exploitation : ici nous avons représenté uniquement les interfaces `IUnknown`, `IDataObject` et `IDispatch` ainsi que la coclasse `DataSource` (qui est utilisée pour faire des transferts de données). La spécification de la coclasse `DataSource` annonce qu'elle fournit les services des deux interfaces `IUnknown` et `IDataObject`.

De plus, nous avons fait figurer une interface supplémentaire, définie par un développeur : `IMonInterface` ainsi qu'une coclasse également définie par un développeur : `MaCoclasse`. Sa spécification annonce qu'elle fournit les services de `IUnknown`, `IDataObject`, `IDispatch` et `IMonInterface`.

Le code exécutable qui met en œuvre ces spécifications est contenu dans le fichier serveur `monprg.exe`, tandis que le serveur de la coclasse `DataSource` est le fichier `datasrc.dll`, qui fait partie du système d'exploitation.

Comment le système d'exploitation est-il informé qu'il dispose à présent d'une interface et d'une coclasse supplémentaire ? Lors de la mise en place du serveur `monprg.exe`, c'est-à-dire lors de sa première utilisation, il vient s'*enregistrer* dans une zone particulière du système d'exploitation : la *base de registre*. Le rôle de cette base est, entre autres, d'expliciter les associations entre les coclasses disponibles et les fichiers exécutables qui sont leurs serveurs.

Par la suite, lorsque l'application `unprogramme.exe` veut utiliser une instance de `MaCoclasse`, elle expose sa requête à la base de registre, qui retrouve le serveur de cette coclasse, en crée une instance et fournit à l'application `unprogramme.exe` une référence (un pointeur) sur cette instance[1].

Attention : la dissociation opérée entre les spécifications des interfaces, celles des coclasses et leurs mises en œuvre fait que différents serveurs peuvent mettre en œuvre différemment une même interface. Par exemple, dans le cas de la figure B.4, les mises en œuvre des méthodes de l'interface `IDataObject` par les serveurs `datasrc.dll` et `monprg.exe` peuvent être complètement différentes. La seule contrainte est qu'elles doivent rendre le même service, remplir le même contrat, mais rien ne les oblige à le remplir de la même façon. (Ceci interdit donc formellement de compter sur tel ou tel effet de bord de la mise en œuvre d'une interface.)

B.1.2.3 L'interface IUnknown

Cette interface permet de résoudre un problème posé par les contraintes d'indépendance des objets COM par rapport à leur langage et à leur compilateur d'origine : puisqu'un objet COM peut comporter un nombre arbitraire d'interfaces, mises en œuvre de façon imprévisible, comment faire pour utiliser un objet COM lorsqu'on a juste une référence sur lui ? De plus, étant donné que plusieurs clients peuvent utiliser une même instance de l'objet, comment faire pour décider quand l'instance doit être détruite ? En effet, si un client, une fois qu'il a fini de l'utiliser, détruit l'instance alors qu'elle est toujours utilisée par un autre client, ce dernier se retrouve en train de "parler dans le vide" (la référence qu'il détient sur l'instance n'est plus valide) et ceci peut avoir des conséquences très fâcheuses.

L'idée est alors de définir une convention : la représentation en mémoire de l'instance possède un en-tête fixe qui contient des offsets de fonctions normalisées qui répondent à ces questions.

La figure B.5 montre le principe utilisé pour cet en-tête. Tout d'abord, il est toujours situé au même endroit par rapport à l'adresse où est rangée l'instance. Ensuite, il contient quatre offsets :

[1] En fait, le processus est plus complexe et plus subtil, mais le schéma de fonctionnement général est celui que nous venons d'exposer.

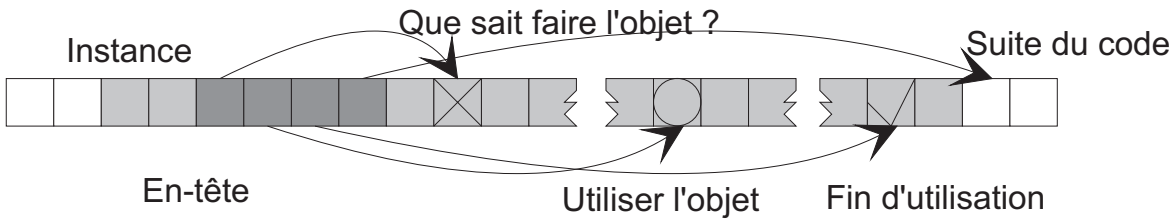

FIG. B.5: Normalisation de l'accès à un objet COM

1. Le premier indique l'adresse d'une fonction qui permet de savoir quels services cet objet peu rendre.
2. Le deuxième indique l'adresse d'une fonction utilisée par le client pour signifier à l'objet qu'il va utiliser ses services (ainsi l'objet sait qu'il possède un client de plus).
3. Le troisième indique l'adresse d'une fonction que le client utilise pour signifier à l'objet qu'il a fini de l'utiliser (et donc l'objet sait qu'il possède un client de moins).
4. Enfin, le dernier offset indique l'adresse de la suite du code exécutable.

Les trois services correspondant aux trois premiers offsets de l'en-tête sont les plus fondamentaux que peut rendre un objet COM. Ils justifient donc à eux seuls la définition d'une interface spécialisée : `IUnknown`. Ce nom, qui pourrait la faire passer pour plutôt anodine, signifie que grâce à elle, on peut manipuler un objet COM dont on ne connaît rien.

Puisqu'elle définit trois services, cette interface contient les spécifications de trois méthodes, qui sont en fait les trois "fonctions spécialisées" indiquées sur la figure B.5 :

1. `IUnknown::QueryInterface` permet d'obtenir une référence sur n'importe laquelle des interfaces fournies par l'objet ;
2. `IUnknown::AddRef` enregistre le client auprès de l'objet qu'il utilise ;
3. `IUnknown::Release` indique à l'objet que le client n'a plus besoin de lui.

Remarquons tout d'abord que le couple `AddRef`/`Release` permet de résoudre le deuxième problème que nous avons exposé plus haut, à savoir quand peut-on détruire une instance ? A supposer que tous les clients d'un objet se comportent correctement, en contrebalançant bien tous leurs `AddRef` par le bon nombre de `Release`, l'objet sait qu'il peut se "suicider" lorsque tous ses clients enregistrés lui ont signalé la fin de leurs transactions.

La méthode `QueryInterface` correspond en fait à la requête « Est-ce que tu offres l'interface `Ixxxx` ? » qu'un client peut soumettre à l'objet. Si non, l'objet renvoie alors un code indiquant l'échec de la requête. Et dans le cas positif, il fournit une référence sur cette interface. Mais que représente exactement cette référence ? En fait, un objet qui offre plusieurs interfaces va simplement empiler plusieurs en-têtes semblables à celui de la figure B.5, chacun comportant une table d'offsets vers le code exécutable des méthodes d'une des interfaces. De plus, on veut, quelque soit l'interface d'un objet qu'on utilise à un moment donné, pouvoir continuer à utiliser les services de `QueryInterface`, `AddRef`

et `Release` (sinon, gare aux accidents !). C'est pourquoi toute interface doit hériter de `IUnknown`, c'est-à-dire que ses trois premières méthodes (i.e. les trois premiers offsets de son en-tête) doivent être (ou plutôt indiquer) une mise en œuvre de `QueryInterface`, `AddRef` et `Release`.

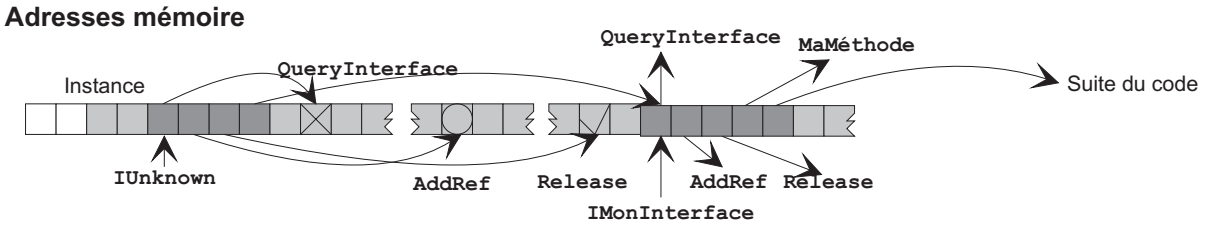

FIG. B.6: Objet COM avec plusieurs interfaces

C'est ce que montre la figure B.6, avec un objet COM offrant au moins les deux interfaces `IUnknown` et `IMonInterface`. On voit que dans l'en-tête correspondant à cette dernière interface, les trois premiers offsets sont similaires à ceux de `IUnknown` (qu'ils indiquent la même adresse ou une autre, voilà qui est laissé à la discrétion du développeur), et le quatrième indique l'adresse de la méthode spécifique à cette interface[1].

Remarque : un appel réussi à `QueryInterface` aura lui-même appelé `AddRef` automatiquement. Il faut tenir compte de cela pour faire le bon nombre d'appels à `Release`.

Si un composant logiciel exporte l'interface `IUnknown` auprès du système d'exploitation, on dit de lui qu'il répond à la norme ActiveX : c'est un contrôle ActiveX. On peut donc imaginer un contrôle n'offrant que cette seule interface. Il serait d'une utilisation peu impressionnante mais pourrait "coopérer" avec tous les autres composants ActiveX du système d'exploitation.

B.1.2.4 Quelques règles d'or

L'adoption du modèle de conception (D)COM implique que les clients et les serveurs doivent s'astreindre à un certain nombre de règles fondamentales.

Les clients d'un objet COM ne doivent interagir avec lui qu'à travers ses interfaces.

Si l'on dispose d'une référence sur un objet COM chargé en mémoire, et étant donné que COM est indépendant du langage de programmation qui a servi à produire cet objet, on n'a aucun moyen de déduire le comportement de l'objet uniquement à partir de la valeur de la référence. En effet, cette référence n'est pas un *pointeur* ordinaire. Il faut donc toujours passer par le protocole des interfaces pour accéder aux méthodes de l'objet.

Chaque interface est un contrat définitif et documenté entre un client et un objet COM qui offre cette interface. Ce contrat ne doit jamais changer.

Le "contrat" défini par une interface n'est susceptible de recevoir aucun amendement dans le futur. Cela implique donc que la documentation concernant l'interface doit être totalement fiable. Cepen-

[1] Cette façon de représenter les interfaces sous forme de tableaux d'offsets est un legs du C++ qui est, rappelons-le, en quelque sorte le "modèle" de COM. C'est ainsi que le C++ organise les interfaces de ses classes. C'est ce qu'on appelle des *tables de méthodes virtuelles*, ou *vtables* en anglais.

dant, cela ne signifie pas que la façon dont le contrat est honoré ne puisse pas changer, seuls les termes sont immuables (cf. la figure B.4 où les serveurs `datasrc.dll` et `monprg.exe` mettent en œuvre différemment l'interface `IDataObject`).

Toute interface doit mettre en œuvre les méthodes de IUnknown pour ses trois premières méthodes.

Cela signifie que, quelque soit "l'angle de vue" sous lequel on approche un objet COM (i.e. quelque soit l'interface de l'objet dont on dispose), il doit être possible à tout moment de référencer, déréférencer l'objet ou d'obtenir une autre de ses interfaces.

Etant donné une référence sur une interface d'un objet COM, il est possible d'obtenir une référence sur n'importe quelle autre interface supportée par cet objet à l'aide de QueryInterface.

C'est un corollaire de la règle précédente. Il suffit donc d'obtenir une fois une référence sur une interface de l'objet pour être ensuite capable de demander à l'objet de nous rendre tous les services qu'il peut fournir.

Un objet COM doit toujours renvoyer la même référence sur sa mise en œuvre de IUnknown à chaque fois que cette interface est explicitement requise.

C'est la propriété d'identité des objets COM : elle permet de comparer deux instances d'une même coclasse. Il n'existe pas de moyen absolu d'adresser une instance particulière d'une coclasse, ce qui constitue une des lacunes majeures de COM.

B.1.3 Création d'une instance

Pour l'instant, nous avons vu comment obtenir n'importe quelle interface d'un objet à partir du moment où l'on dispose déjà au moins d'une référence sur sa mise en œuvre de `IUnknown`. Mais, justement, comment obtenir cette référence ? C'est ici que le schéma de comportement générique de COM se lézarde : le processus de création d'une instance d'une coclasse est dépendant à la fois de la machine et de son système d'exploitation.

Tout d'abord, il faut pouvoir repérer la coclasse dont on veut créer une instance. Cette coclasse, depuis sa mise en service, fait partie du système d'exploitation de la machine, comme nous l'avons vu sur la figure B.4. Mais il faut aussi prendre en compte le cas où la création d'une instance est demandée par un client situé sur une machine distante. La solution typique dans ce genre de problème est de coordonner les noms des entités (coclasses et interfaces) grâce à une sorte d'autorité centrale, de façon similaire à ce qui se passe pour les adresses IP d'Internet.

B.1.3.1 Les GUID

La solution proposée par l'OSF (*Open Software Foundation*) est d'utiliser un "identificateur universellement unique", ou UUID (*Universally Unique Identifier*). Microsoft a opté pour une version moins ambitieuse : l'identificateur globalement unique, ou GUID (*Globally Unique Identifier*). Il s'agit d'un nombre

de 128 bits basé sur la date et l'heure de sa production, l'adresse IP de la machine qui l'a engendré ainsi que le numéro de série de sa carte mère, et un nombre pseudo-aléatoire.

Il existe tellement peu de chances que deux GUID entrent en collision que presque tous les composants logiciels peuvent recevoir un GUID unique pendant encore de nombreuses années. En fait, tout ce qui a trait au modèle COM reçoit effectivement un GUID : les GUID pour les coclasses sont appelés CLSID (*Class ID*) ; pour les interfaces, ce sont des IID (*Interface ID*) ; pour les bibliothèques de types[1], ce sont des LIBID (*Library ID*), etc.

On va alors pouvoir utiliser ces identificateurs quasi uniques pour la coordination des objets à travers un réseau de taille arbitrairement large. Il suffit de créer sur chaque machine, au sein du système d'exploitation, une base de données indexée par les GUID. Chaque entrée de cette base décrit un composant logiciel reconnu par la machine (et donc utilisable par un client) en fournissant son GUID, son nom (sous forme lisible) relativement à cette machine, ainsi que diverses informations sur la façon d'invoquer ce composant (quel programme exécutable en est le serveur, à quelle adresse peut-on le trouver, ...) L'utilisation du GUID comme clé, plutôt que le nom du composant, permet que deux personnes puissent développer des composants différents portant le même nom, sans qu'il y ait de collision lorsqu'ils sont invoqués depuis un autre poste de travail[2].

Cette base de donnée est ce que nous avons nommé la "base de registre" sur la figure B.4. Elle est mise à jour à chaque installation (mise en service) ou désinstallation d'un composant logiciel dans le système d'exploitation.

B.1.3.2 Les Class Factories

Pour l'instant, grâce à la base de registre, nous savons comment retrouver le programme serveur d'un composant logiciel à partir de son GUID, ou de son nom local sur une machine. Mais il reste encore à invoquer ce serveur pour lui demander de créer une instance du composant logiciel et de fournir en retour une référence sur cette instance. En termes de programmation orientée objets, il nous reste à savoir comment on appelle le constructeur d'une coclasse.

A ce niveau, le modèle COM se distingue à nouveau de la programmation orientée objets traditionnelle : **le constructeur d'une coclasse est un objet COM indépendant**. Cet objet particulier porte le nom de *class factory*, ce qui est bien mal venu puisque son rôle est de construire des instances, et non pas des classes. Chaque *class factory* est une instance d'une coclasse exposant comme unique interface `IClassFactory`, et qui est spécialisée dans la construction d'instances d'une coclasse particulière (ceci est bien évidemment à la charge du concepteur de la coclasse).

Ainsi, si un serveur met en œuvre un certain nombre de coclasses `classeA`, `classeB`, ... il devra contenir une instance de *class factory* spécialisée pour chacune de ses coclasses. Ces "usines à instances" appartiennent au serveur, et ne font pas partie du composant logiciel qu'elles sont chargées de construire.

Cette organisation un peu déroutante est due à la question : « Qui construit les constructeurs ? » En effet, puisqu'une *class factory* est une instance d'un objet COM, il faudrait, avant de pouvoir l'utiliser, d'abord la construire, et donc accéder à sa propre *class factory*. Cela impliquerait une récursion infinie, et le problème a été résolu en brisant l'indépendance de COM vis à vis du système d'exploitation : la construction d'une *class factory* est assurée par le système d'exploitation lui-même lorsqu'il charge en mémoire le serveur correspondant.

[1] Que nous verrons plus loin.

[2] En effet, on ne peut pas demander à l'ensemble de la communauté des programmeurs de coordonner les noms qu'ils donnent à leur programmes.

L'accès à la *class factory* d'une coclasse se fait à l'aide d'une API[1], c'est-à-dire une fonction standard du système d'exploitation : c'est la seule occasion pour laquelle le modèle COM dépend de son système d'exploitation hôte. Cette API, `CoGetClassObject`, renvoie une référence sur la *class factory* chargée de construire la coclasse dont le CLSID a été mentionné en paramètre. On peut alors utiliser les méthode de l'interface `IClassFactory` exposées par cet objet afin de créer une instance de la coclasse désirée.

De façon usuelle, on utilisera plutôt une autre API, `CoCreateInstance`, qui réalise ces deux opérations en une seule fois :

1. obtenir la *class factory* attachée à un CLSID grâce à `CoGetClassObject`, puis
2. créer une instance de cette coclasse et renvoyer une référence sur l'objet nouvellement créé.

B.1.4 Utilisation d'un objet COM

Si l'on a créé une instance d'un objet COM, et qu'on dispose donc d'un pointeur sur une de ses interfaces, comment faire pour l'utiliser ? Puisqu'une interface est un contrat (une spécification), les méthodes de cette interface doivent être documentées quelque part afin d'en connaître le prototype. L'utilisateur disposant d'une référence (il s'agit la plupart du temps d'un pointeur, pour les langages qui le permettent) sur l'interface `Ixxxx` peut invoquer ses méthodes exactement comme s'il s'agissait d'une classe normale d'un langage orienté objet.

Par exemple, pour invoquer la méthode `Method` de l'interface `Ixxxx` sur un pointeur `pIxxxx`, un programmeur C++ écrira `pIxxxx->Method(...)` et un programmeur Pascal écrira plutôt `pIxxxx↑.Method(...)`.

Dans la pratique, il faut cependant disposer de fichiers d'en-tête contenant les déclarations des constantes et les profils des méthodes des interfaces requises. Les pointeurs sur les méthodes sont alors convertis par le compilateur et l'éditeur de liens en décalages (*offsets*) dans l'image exécutable de l'objet, ce qui a pour effet d'engendrer un overhead équivalent à l'appel d'une méthode d'une classe normale d'un langage orienté objets. Cette approche est généralement appelée *very early binding* (liaison très précoce) ou *vtable binding*. Elle permet au compilateur de préparer la meilleure façon "d'attacher" l'objet COM dans son code exécutable.

Cependant, dans de nombreux cas, il est impossible (ou trop limitatif) de réaliser une liaison précoce, par exemple, lorsqu'un objet COM est utilisé par un interpréteur (Visual Basic est le langage interprété le plus répandu sur les plates-formes WINDOWS). Ce type de langage ne peut avoir recours aux déclarations des interfaces sous forme de fichiers d'en-tête. Or, le code exécutable de l'interpréteur ne peut connaître à l'avance tous les objets COM qu'il pourra un jour utiliser, sinon, à chaque fois qu'un nouvel objet COM est développé, il faudrait recompiler et redistribuer l'interpréteur.

Il faut donc pouvoir réaliser une *liaison tardive* : l'application ne saura quoi faire de l'objet qu'au cours de son exécution. Il existe dans le système d'exploitation une interface spécialement dédiée à l'exploration d'interfaces inconnues et à leur liaison tardive : `IDispatch`. L'application cliente devra donc

[1] *Application Programming Interface*

effectuer une liaison précoce avec `IDispatch`, qui l'aidera en retour à effectuer une liaison tardive sur les interfaces encore inconnues en cours d'exécution. Cependant, cela ne résout toujours pas le problème de la "sémantique" des méthodes de l'interface qui a été liée grâce à `IDispatch`.

Pour résoudre ce problème, on est obligé de forcer toutes les interfaces d'objets COM à se couler dans un moule "object-like" : les coclasses qui exposent l'interface `IDispatch` peuvent mettre en œuvre un ensemble de propriétés (semblables aux variables membres d'une classe normale) auxquelles on peut accéder par des méthodes `Get` et `Set`, ainsi qu'un ensemble de méthodes (semblables aux fonctions membres d'une classe) qu'on peut invoquer à travers l'interface `IDispatch`.

Cela a pour effet d'introduire une nouvelle couche d'indirection pour accéder aux fonctionnalités de l'objet COM. Cependant, ce que l'on perd en efficacité, on le gagne en souplesse d'utilisation. N'importe quel objet COM exposant l'interface `IDispatch` peut être utilisé par n'importe quelle application liée à l'interface `IDispatch` sans que les deux aient à se connaître préalablement.

B.1.5 Les relations entre coclasses

Nous avons au vu au §B.1.2.1 que les classes d'un langage orienté objets à base de classes peuvent entretenir des relations ontologiques (c'est la relation d'héritage, aussi appelée *ako*, pour "a kind of"), méréologiques (il s'agit de la relation de composition, ou *apo*, pour "a part of") ou enfin d'instanciation (c'est la relation qui existe entre un objet et sa classe, ou relation *isa*, pour "is a"). Dans le modèle (D)COM, si la relation d'instanciation est maintenue, l'héritage et la composition ont été remplacées par deux autres relations : la *délégation* et l'*agrégation*.

Il faut garder à l'esprit que dans le cas de (D)COM, il ne s'agit plus de réutiliser des classes au sein d'un programme, c'est-à-dire d'inclure ou pas des portions de programme dans un autre, mais de réutiliser des composants logiciels exécutables, et par conséquent opaques à leurs clients, sauf en ce qui concerne les interfaces.

Pour illustrer les deux paragraphes qui vont suivre, nous adopterons la convention graphique en vigueur dans la littérature concernant COM et ActiveX. La figure suivante représente un objet COM atomique, c'est-à-dire n'exportant que l'interface `IUnknown`. Les interfaces sont représentées graphiquement par des "fiches de jack" (comme en électronique), car l'allégorie veut qu'un client utilise un objet COM en "enfichant" une connexion dans une de ses interfaces. Traditionnellement, l'interface `IUnknown` est située au sommet de l'objet, tandis que les autres interfaces sont représentées sur les côtés.

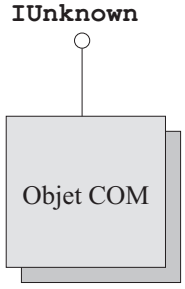

FIG. B.7: Représentation d'un objet COM

B.1.5.1 Inclusion d'objet et délégation d'interface

La façon la plus évidente de réutiliser des objets COM existant dans un nouveau projet, est d'en créer des instances qui seront utilisées par un nouvel objet COM. Ce processus s'appelle *inclusion d'objet*, puisque le nouvel objet est considéré par le système d'exploitation comme incluant les instances des objets qu'il utilise, ce qui est similaire à la relation de composition exposée plus haut.

Le fait d'invoquer une méthode d'un objet inclus en réponse à l'invocation d'une méthode de l'objet conteneur porte le nom de délégation, étant donné que le conteneur délègue le travail, ou une partie du travail, à l'une des instances qu'il inclut.

Dans l'exemple de la figure B.8, on dispose de deux objets COM déjà écrits : un agent Acheteur de viande et un agent Acheteur de pain. On veut maintenant écrire un nouvel agent qui fait des achats d'alimentation pour ses clients.

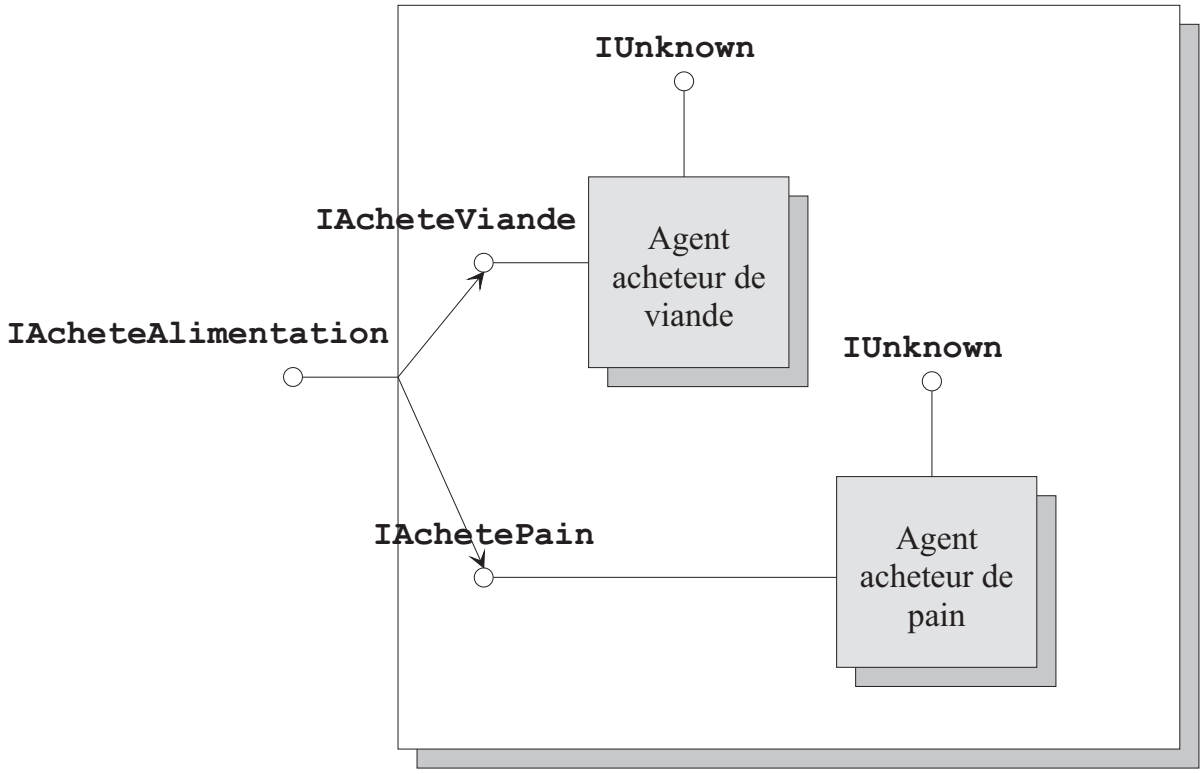

FIG. B.8: Exemple de délégation

Lorsqu'un client de l'agent Acheteur de nourriture invoque la méthode `IAcheteAlimentation::FaireLesCourses`, l'agent délègue la tâche aux instances qu'il contient en invoquant les méthodes de `IAcheteViande` et `IAchetePain`. Cet objet pourra lui-même être inclus dans un autre objet ultérieurement.

B.1.5.2 Agrégation

Dans le cadre de l'inclusion, l'objet conteneur s'interpose entre ses clients et les instances des objets qu'il inclut. Mais on peut aussi se retrouver dans la situation où un client serait tout à fait capable d'utiliser lui-même les interfaces d'une instance incluse. L'objet conteneur ne jouerait alors qu'un rôle

“d’emballage” réduisant l’efficacité du code sans apporter de valeur ajoutée. L’alternative à l’inclusion est alors l’*agrégation* : l’objet conteneur expose directement les interfaces des instances qu’il contient.

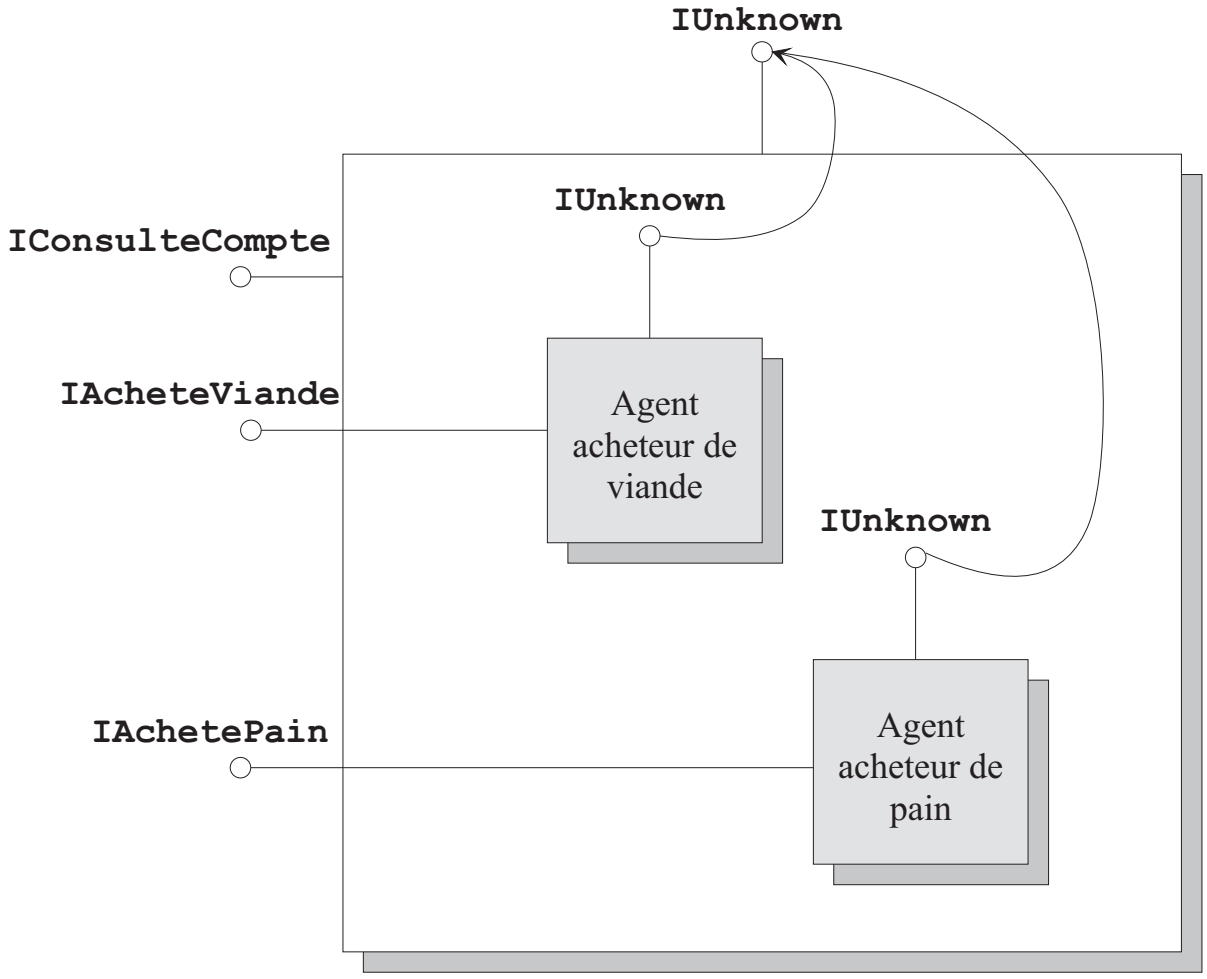

FIG. B.9: Exemple d’agrégation

Vu de l’extérieur, l’agent Acheteur de nourriture expose les trois interfaces `IConsulteCompte`, `IAcheteViande` et `IAchetePain` et ses clients n’ont aucun moyen de savoir qu’il utilise en fait deux instances incluses pour effectuer le travail des deux dernières interfaces.

Sur la figure B.9, des pointeurs réfèrent à l’interface `IUnknown` de l’objet conteneur depuis les interfaces `IUnknown` des objets contenus. Ceci pour deux raisons :

— Tout d’abord, comme précisé au §B.1.2.4, un objet COM doit toujours fournir le même pointeur sur `IUnknown` à chaque fois que cette interface est requise explicitement par l’intermédiaire de la méthode `QueryInterface` de n’importe quelle interface. En effet, si un client de l’objet conteneur obtient un pointeur sur son interface `IConsulteCompte` ainsi qu’un pointeur sur son interface `IAcheteViande`, et sans ce transfert de pointeurs, le résultat de `IConsulteCompte::QueryInterface` pour `IUnknown` serait différent du résultat de `IAcheteViande::QueryInterface` pour `IUnknown`, puisqu’on obtiendrait dans un cas le pointeur sur l’interface `IUnknown` de l’objet conteneur, et dans l’autre le pointeur sur `IUnknown` de l’instance de l’agent Acheteur de viande.

— En corollaire de ce premier point, il ne serait plus possible d'obtenir n'importe quelle interface de l'objet conteneur à partir de n'importe quelle autre interface de départ (puisque l'agent Acheteur de viande n'exporte pas l'interface `IConsulteCompte`). De plus, il serait impossible de vérifier l'unicité d'une instance de l'objet conteneur à partir de ses différentes interfaces, puisque les pointeurs sur `IUnknown` seraient différents.

Ce phénomène d'agrégation, tout comme la délégation, peut se répéter plusieurs fois. A chaque étape, c'est le pointeur sur `IUnknown` de l'objet le plus extérieur qui est passé aux instances intérieures, puisque du point de vue du système, seul l'objet le plus extérieur est informé de son état d'agrégation.

En conclusion, le couple de relations délégation/agrégation se comporte d'une façon extrêmement similaire au couple composition/héritage (pour une plus longue discussion sur cette assertion, on peut se reporter à Li et Economopoulos [67, pp. 69-70]).

Ces relations offrent pourtant une marge de manœuvre plus réduite en ce qui concerne la surcharge des méthodes héritées entre classes, puisqu'ici, ce sont les interfaces entières qui sont agrégées ou pas. Ceci est dû à l'immuabilité des interfaces dans COM (cf. les règles d'or, p. 342). En contrepartie, tant que les contrats des interfaces sont respectés, un objet agrégé peut être modifié, se voir ajouter ou retrancher des interfaces, sans que ses clients en soient affectés. Le système est donc moins souple, mais plus robuste.

B.2 Automation

Nous avons vu au §B.1.4 que l'utilisation d'une instance d'un objet COM peut se faire soit grâce à une liaison précoce aux méthodes de ses interfaces, soit par une liaison tardive, grâce à une liaison précoce à l'interface `IDispatch`.

L'utilisation la plus intensive qui est faite de l'interface `IDispatch` est l'automatisation de tâches au sein d'une application grâce à un langage permettant l'accès aux objets COM supportant `IDispatch`. Parmi les langages les plus répandus actuellement, viennent en tête les produits de Microsoft : Visual Basic et Visual Basic pour Applications, mais aussi PERL (issu du monde Unix) et VBScript (qui permet d'écrire des *applets* Internet, comme Java, mais recourant au seul ActiveX). La popularité toujours grandissante de ces outils à conduit Microsoft à donner un nom séparé à ce mécanisme de liaison tardive : *Automation* (pour automatisation).

B.2.1 L'interface IDispatch

Nous n'entrerons pas ici dans les détails de l'interface `IDispatch` ; pour plus de précisions, on peut se reporter aux ouvrages de Li et Economopoulos [67], ainsi qu'à celui de Brockschmidt [15]. Il suffira de dire qu'elle expose trois méthodes fondamentales pour utiliser un objet COM alors qu'on ne connaît pas encore tous les offsets de ses méthodes (cf. §B.1.2.3) : `GetTypeInfo`, `GetIDsOfNames` et `Invoke`.

Ces méthodes s'utilisent normalement dans l'ordre où elles ont été citées. Elles correspondent respectivement aux tâches suivantes :

— `IDispatch::GetTypeInfo` sert à "consulter la documentation" concernant une méthode ou une propriété. Cette documentation est contenue, pour chaque serveur COM, dans un fichier spécial que l'on appelle *bibliothèque de types* (cf. §B.2.1.2).

— `IDispatch::GetIDsOfNames` permet d'obtenir un identificateur spécial à partir du nom d'une méthode, et qui va servir à calculer automatiquement l'offset de cette méthode.

— `IDispatch::Invoke` effectue l'invocation d'une méthode ou d'une propriété, une fois que l'on connaît ses contraintes de typage (obtenues grâce à `GetTypeInfo`) et son identificateur (obtenu par `GetIDsOfNames`).

B.2.1.1 Les identificateurs DispID

Automation prévoit un certain nombre de degrés de liberté pour ses utilisateurs, notamment la possibilité de donner des noms différents aux propriétés, méthodes et arguments d'une interface en fonction de la langue parlée par l'utilisateur ; également la possibilité de changer l'ordre des arguments d'une méthode lors de son invocation en les repérant par leur nom plutôt que par leur place (comme avec le langage ADA).

Pour repérer la portion de code exécutable qui correspond à une propriété, une méthode ou un argument, quelque soit son nom local et sa position, Automation utilise des identificateurs numériques, les *DispIDs* qui sont uniques pour chaque élément à l'intérieur de sa zone de portée (à l'intérieur d'une méthode pour un argument et à l'intérieur d'une interface pour une méthode).

Le rôle de la méthode `IDispatch::GetIDsOfNames` est de retrouver le DispID d'un élément à partir de son nom et de son contexte d'utilisation.

B.2.1.2 Les bibliothèques de types

Comme nous l'avons dit plus haut (cf. §B.1.4), l'exploration d'une interface inconnue à l'aide du mécanisme de liaison tardive pose un problème car on ne dispose pas de la "sémantique" des éléments de l'interface. Pour résoudre ce problème, on utilise un fichier spécial, une *bibliothèque de types*, qui décrit très précisément les spécifications de chaque élément (coclasses, interfaces, propriétés et méthodes ainsi que leurs arguments). Ces spécifications permettent de connaître pour chaque élément son nom, le DispID qui lui est associé, et pour les propriétés et méthodes, quels arguments elles attendent, ainsi que leur type et leur statut (entrée et/ou sortie).

La consultation de la bibliothèque de type associée à un serveur permet donc de vérifier que les requêtes qui lui sont soumises "signifient" quelque chose pour lui.

> Les bibliothèques de types peuvent être manipulées par des requêtes à des objets COM spécialisés, possédant leurs propres interfaces dédiées à ces manipulations. Mais on peut aussi beaucoup plus facilement les écrire et les modifier sous forme de fichiers texte. Microsoft fournit un compilateur pour une version étendue du langage standard IDL (pour *Interface Definition Language*) de DCE[1]. Ce compilateur va lire le fichier texte et engendrer les appels nécessaires à COM pour créer la bibliothèque de types correspondante.

L'utilisation la plus courante qu'un utilisateur peut faire d'une bibliothèque consiste à la charger en mémoire par l'intermédiaire de l'API `LoadTypeLib`, ce qui a pour effet de créer les objets COM spécialisés dans sa manipulation, et de renvoyer un pointeur sur une interface spéciale, `ITypeLib` pour de plus amples requêtes à propos de la bibliothèque.

Une bibliothèque de types décrit en détail à peu près tout ce que contient et utilise un objet COM donné, c'est-à-dire :

— des coclasses ;

— des interfaces, qu'elles soient standard (fournies originellement avec COM), personnelles (définies par des développeurs), dérivées de `IDispatch` (ce que nous venons de voir), ou duales (ce que nous verrons dans le prochain paragraphe) ;

— des propriétés et des méthodes `IDispatch` ;

— des définitions de types classiques (comme en C++ ou en Pascal) et des constantes.

Une fois compilée, une bibliothèque de types peut soit être placée dans un fichier spécial d'extension .TLB, ou bien être incorporée dans le fichier contenant l'exécutable du serveur COM qu'elle décrit. De plus, lors de sa mise en place (sa première utilisation), elle doit être enregistrée dans la base de registre.

B.2.2 Les interfaces duales

Jusqu'ici, il fallait choisir, lorsqu'on développait un objet COM, entre offrir une liaison précoce ou offrir une liaison tardive avec `IDispatch`. Nous allons maintenant voir comment offrir ces deux possibilités simultanément.

Ceci est particulièrement important pour certains environnements, comme Visual Basic, qui utilisent la liaison tardive lorsqu'ils fonctionnent en mode interpréteur, et cherchent à obtenir des liaisons précoces en mode compilateur, pour des raisons d'optimisation. De plus, certains langages comme PERL ou VBScript recourent uniquement à la liaison tardive, tandis que d'autres, tels Visual C++ ou Delphi, sont plutôt conçus pour utiliser des liaisons précoces.

Une interface offrant les deux possibilités de liaison s'appelle une interface *duale*, et est marquée du mot-clé `dual` dans son fichier de définition IDL. Elle dérive automatiquement de l'interface `IDispatch`, et tous ses membres doivent impérativement fournir un code d'erreur spécial comme valeur de retour. C'est en effet la façon dont COM modélise les exceptions : ce code d'erreur est un entier de 32 bits répondant à certaines règles d'interprétation qui permettent de déterminer s'il s'agit d'un succès ou d'une erreur, et le cas échéant, la gravité et la nature de l'erreur.

Ainsi, lorsqu'on définit une interface duale, il faut fournir la mise en œuvre des méthodes et des propriétés faisant partie de la partie personnalisée de l'interface, mais aussi la mise en œuvre des méthodes de l'interface `IDispatch` (en raison de l'héritage obligatoire des méthodes de `IDispatch`).

B.2.3 Les types standards d'Automation

Les spécifications d'Automation prévoient l'emploi d'un certain nombre de types de données standards pour les transactions entre clients et serveurs. Bien entendu, il est tout à fait possible de définir de nouveaux types personnalisés.

Ces types standards sont ceux qu'on est en droit d'attendre avec tout langage de programmation traditionnel, notamment différentes catégories d'entiers et de nombres à virgule flottante, ainsi que des

types pointeurs. Mais à ces types se rajoutent quelques types particuliers, dus aux applications principales d'Automation, c'est-à-dire la bureautique : un type date, un type monétaire.

On notera aussi trois types "exotiques" :

— un type permettant de manipuler des tableaux contenant, outre des données, des informations d'allocation et de verrouillage en mémoire (afin de sécuriser les transactions de données à travers les réseaux) ;

— un type destiné à normaliser la représentation des chaînes de caractères (`BSTR`), puisqu'Automation, étant basé sur COM, se doit d'être indépendant du langage de programmation employé ;

— un type permettant de représenter une valeur de n'importe quel autre type standard (`VARIANT`), y compris la valeur "vide" afin de gérer l'omission de paramètres lors de l'invocation d'une méthode.

B.2.4 Les collections au sens de Visual Basic

L'interface `IDispatch` (ou une interface duale héritant de `IDispatch`) doit exposer au moins les trois méthodes / propriétés suivantes pour être considérée comme une collection :

1. `long Count` : propriété en lecture seule qui renvoie le nombre d'éléments contenus dans la collection ;
2. `Ixxx* Item(<unType> index)` : propriété en lecture seule fournissant une référence à un des éléments de la collection repéré par son index. Elle renvoie en fait un pointeur sur l'interface `IDipatch` de l'élément, ou bien sur une des interfaces duales de l'élément (cette interface `Ixxx` est fixée lors de la définition du profil de `Item`). Le type de l'index est laissé à l'appréciation du concepteur de `Item`. On notera que la propriété `Item` doit avoir un DispID de 0 dans son interface pour être considérée comme "méthode par défaut" par Visual Basic. Dans ce cas, pour une instance de collection `coll`, on écrira indifféremment en Visual Basic `coll.Item[i]` ou `coll[i]`.
3. La dernière propriété requise est un peu particulière, et c'est pourquoi nous donnons plutôt son profil IDL :

   ```
   [propget, id(-4), restricted]
   HRESULT _NewEnum([out, retval] LPUNKNOWN* pVal);
   ```

   Il s'agit donc d'une propriété en lecture seule (il n'y a pas de définition correspondante marquée de l'attribut `propput`) et qui porte obligatoirement le DispID -4 au sein de son interface. L'attribut `restricted` signifie qu'elle n'est pas accessible depuis un langage de programmation qui serait un client Automation utilisant l'interface `IDispatch`, ce qui revient à dire qu'aucun langage habituel utilisant Automation, tel que Visual Basic, PERL, etc., ne peut l'invoquer. De plus, le caractère souligné en préfixe de son nom lui confère implicitement un attribut `hidden`, qui empêche cette propriété d'apparaître dans les outils d'exploration dynamique des bibliothèques de types en cours d'exécution. Tout ceci peut en fait se résumer à : l'utilisateur d'un langage de programmation Automation, même doté d'une interface d'assistance à la programmation, ne peut pas accéder à cette méthode. La seule utilisation est qu'elle est nécessaire si l'on veut que la collection supporte l'instruction itérative `For Each ... Next` du Visual Basic. Cette instruction permet d'énumérer tous les éléments contenus par une collection sans se préoccuper de leur index. Ainsi, pour une collection `coll`, on pourra écrire

`For Each x In coll` <instructions> `Next x`

Quoiqu'on puisse penser de l'utilité d'une telle propriété, elle est requise par Visual Basic pour les collections. Son rôle est en fait de construire une instance d'un objet COM spécial, appelé *énumérateur*, en lui fournissant, dans un `VARIANT`, un pointeur sur l'interface `IDispatch` de chacun des éléments qu'elle contient. Elle renvoie ensuite un pointeur sur l'interface `IUnknown` de l'énumérateur. C'est ce pointeur que récupère l'instruction `For Each`, et qui s'en sert pour obtenir un pointeur sur l'interface `IEnumVARIANT` de l'énumérateur. Cette interface possède les méthode nécessaires pour énumérer la collection.

Une collection peut, bien entendu, exposer d'autres méthodes et propriétés, par exemple pour l'ajout ou le retrait d'éléments. Si les éléments qu'elle contient sont des propriétés étendues, elle contiendra notamment une méthode `Add` et une méthode `Remove` qui feront office respectivement de constructeur et de destructeur pour ces éléments (N.B. une propriété étendue n'est pas instanciable directement, car on ne peut pas utiliser sa *class factory*).

B.3 La partie immergée de l'iceberg

Ce qui a été présenté dans les §B.1 et §B.2 constitue les fondements de la technologie ActiveX, mais n'en représente qu'une partie assez réduite. Nous nous sommes contenté d'exposer ici les éléments d'ActiveX qui nous ont servi à la mise en œuvre du modèle PELEAS.

La technologie ActiveX offre également les fonctionnalités suivantes, que nous ne détaillerons pas ici (on pourra se reporter plutôt à [15]) :

— *les interfaces de notification* : ces interfaces fonctionnent "à l'envers" des interfaces que nous avons vues jusqu'ici. Elles permettent à un objet COM serveur d'envoyer des messages et des requêtes à ses clients.

— *les "objets de données" (data objects)* : ils forment une utilisation particulière des interfaces de notification. Ils permettent à deux objets COM d'échanger des données sous un format universel et qui est toujours le mieux adapté possible à l'application qui les utilise. Ils permettent également à un serveur de prévenir ses clients que les données qu'il détient (et qu'ils sont susceptibles d'utiliser) ont été modifiées.

— *le Presse-papiers OLE* : est une mise en œuvre des fonctionnalités habituelles d'un presse-papiers (copier, couper, coller) à l'aide des *data objects* afin d'optimiser leur efficacité.

— *le Drag-and-Drop OLE* : est une collection d'objets COM offrant des services de transfert de données (grâce aux *data objects*) visuels entre les représentations à l'écran de deux objets COM.

— *les fichiers structurés* : ils permettent de structurer les données dans des "fichiers à l'intérieur du fichier".

— *les monikers* : ce sont des objets COM qui sont l'équivalent de pointeurs persistants sur d'autres objets COM (on peut notamment les enregistrer dans un fichier, et les recharger ultérieurement en conservant leur validité).

— *les interfaces de visualisation* : elles permettent à un objet COM d'incorporer ses propres méthodes d'affichage standard.

— *la technologie "In Place"* : il s'agit d'un ensemble d'interfaces permettant à un objet COM d'être affiché et modifié à l'intérieur de la représentation à l'écran d'un objet conteneur. Les éléments d'interface graphiques nécessaires à sa modification sont ajoutés et supprimés dynamiquement de l'interface graphique du conteneur suivant les besoins.

— *les documents OLE* : ce sont des objets COM bénéficiant de la technologie "In Place". Un document OLE peut notamment contenir d'autres documents OLE. Cela permet de manipuler des données diverses (textes, feuilles de calculs, base de données, etc.) dans un seul document, à partir d'une seule application, l'exécution des applications spécifiques à chaque source de données du document étant dynamique et transparente pour l'utilisateur.

— *les contrôles ActiveX* : ils font appel à tous les éléments cités ci-dessus, plus d'autres. Ce sont en fait des sortes de documents OLE qui contiennent leur propre application spécifique et qui peuvent être distribués sur un Intranet ou sur Internet de façon à être autosuffisants.

Annexe C

Fonctionnement des DLL

Historiquement, les DLL ont été conçues comme une extension du principe des bibliothèques de code et de données classiquement utilisées par les compilateurs. Pour simplifier l'exposé, nous n'envisagerons que le cas de bibliothèques contenant du code uniquement.

C.1 Principe des bibliothèques dynamiques

Une bibliothèque "traditionnelle" contient des fonctions sous une forme binaire propre au compilateur qui l'utilise, et est dite statique. Lors de l'édition de liens, toutes les fonctions de cette bibliothèque qui sont appelées au sein du programme en cours de compilation, et celles-là seulement, sont intégrées dans le fichier exécutable qui est le résultat de la compilation. Ce schéma présente trois inconvénients pour lesquels les DLL apportent une solution :

— Le format binaire des bibliothèques statiques est spécifique à chaque compilateur ; ces bibliothèques ne peuvent donc être échangées entre différents environnements de développement.

— L'édition de liens étant statique, toutes les fonctions d'une bibliothèque qu'on souhaite utiliser doivent être connues lors de l'écriture du programme.

— Deux programmes faisant appel aux mêmes fonctions d'une même bibliothèque statique possèdent chacun une copie de ces fonctions dans leur exécutable. Par conséquent, lorsque ces programmes sont chargés simultanément en mémoire, il y a duplication de code.

Les DLL répondent à ces restrictions de la manière suivante :

— Le format des DLL est standardisé par le système d'exploitation (c'est en fait le même que celui des fichiers exécutables), ainsi tout environnement de développement peut faire appel à une DLL, sans contrainte de langage ou de plate-forme.

— L'édition de lien est dynamique. Un programme client peut donc décider de quelles DLL il a besoin, et de quelles fonctions au sein de ces DLL en cours d'exécution, sans que ce choix ait été imposé d'avance. Un autre avantage de l'édition de lien dynamique est que la mise en œuvre d'une DLL peut changer sans qu'il soit besoin de toucher au code des programmes clients, pour peu qu'elle conserve les mêmes fonctions exportées.

— Une DLL n'est chargée qu'une seule fois en mémoire à un moment donné. Si deux programmes clients de cette DLL sont chargés simultanément, la DLL est simplement *mappée* dans les espaces d'adressage de ces deux clients. C'est-à-dire que les adresses réelles où sont stockés les différents éléments de la DLL sont converties en adresses virtuelles de l'espace d'adressage de chaque client. Ainsi, ils ont tous deux l'impression que les fonctions exportées par la DLL font désormais partie de leurs processus respectifs et que chacun possède donc une copie indépendante de la DLL (en fait, si les adresses des fonctions sont ainsi virtualisées, ce n'est pas le cas des données locales de la DLL : chaque client possède une instance propre de ces données).

C.2 Mécanisme d'édition de liens dynamique

Un client, que ce soit une application ou une DLL, amorce l'édition de liens dynamique en chargeant une DLL, ce qui lui permet d'obtenir une instance de cette DLL. Toute DLL exporte au moins une fonction particulière, appelée *point d'entrée* de la DLL. La création d'une instance de la DLL a pour conséquence d'invoquer son point d'entrée avec un paramètre indiquant l'attachement à un processus ; c'est le moment privilégié pour initialiser les éventuelles données statiques de la DLL. Ceci fait, le système d'exploitation incrémente un compteur de références sur la DLL, signifiant qu'un processus supplémentaire y fait appel.

L'édition d'un lien avec une fonction exportée par la DLL est réalisé par le *mappage* explicite de l'adresse de cette fonction dans l'espace d'adressage du client. Ceci permet d'obtenir un pointeur sur la fonction qui est valide dans l'espace d'adressage du client. Il faut cependant conserver à l'esprit que le pointeur ainsi obtenu est un *pointeur de fonction générique*, et qu'il convient de le transtyper convenablement avant de l'utiliser, ne serait ce que pour lui faire correspondre le bon profil de fonction.

Lorsque le client n'a plus besoin des services de la DLL, il doit la décharger. Ceci a pour effet, en premier lieu, d'invoquer à nouveau le point d'entrée, avec un paramètre indiquant son détachement du processus, ce qui signifie qu'il est temps de finaliser et de nettoyer les données statiques de la DLL. En second lieu, le mappage de la DLL dans l'espace d'adressage du client est supprimé, et donc les pointeurs obtenus précédemment par l'édition de liens manuelle ne sont plus valides. Finalement, le système d'exploitation décrémente le compteur de références sur cette DLL. Si ce compteur arrive à 0, la DLL est déchargée de la mémoire.

Le chargement d'une DLL, l'édition de lien dynamique d'une fonction et le déchargement de la DLL sont réalisés par les fonctions de l'API Win32 suivantes :

Chargement
```
HINSTANCE LoadLibrary(char* szDllFilename)
```
Edition de lien
```
FARPROC   GetProcAdress(HINSTANCE hLib,
                        char* szFuncName)
```
Déchargement
```
void      FreeLibrary(HINSTANCE hLib)
```

Le point d'entrée d'une DLL s'appelle `DllMain` par défaut, mais son nom peut être modifié par l'auteur de la DLL. S'il n'est pas fourni lors de l'écriture de la DLL, le compilateur ajoute automatiquement un point d'entrée qui n'exécute aucune action. Le profil d'un point d'entrée est :

```
BOOL DllMain(HINSTANCE hLib,
             DWORD dwReason,
             LPVOID lpvReserved)
```

Lors du chargement de la DLL, le point d'entrée est appelé avec le paramètre `dwReason` fixé à `DLL_PROCESS_ATTACH`, et lors du déchargement il est fixé à `DLL_PROCESS_DETACH`. La DLL peut aussi gérer le multitâche de façon fine, car le point d'entrée est appelé avec `dwReason` fixé à `DLL_THREAD_ATTACH` lors de son attachement à un nouveau chemin d'exécution, et à `DLL_THREAD_DETACH` lorsque le chemin d'exécution libère la DLL.

Annexe D

Spécifications formelles

D.1 Spécifications des coclasses de LIGHTPELEAS

D.1.1 Spécification de la coclasse Engine

SPEC : ENGINE ;
USE : PATH, ENTRIES ;
SORT : Engine ;
"La spécification PATH décrit les noms de fichiers et leurs chemins. Cela nous permet d'utiliser des prédicats tels que FileExists pour déterminer si un fichier existe."

GENERATED BY :
CreateInstance : $\rightarrow$ Engine ;

OPERATIONS :
EntriesDirectory _ : Engine $\rightarrow$ Path ;
EntriesDirectory _ _ : Engine Path $\rightarrow$ Engine ;
IndexFile _ : Engine $\rightarrow$ Path ;
IndexFile _ _ : Engine Path $\rightarrow$ Engine ;
GenerateIndex _ : Engine $\rightarrow$ Engine ;
GenerateIndex _ _ : Engine Path $\rightarrow$ Engine ;
Entries _ : Engine $\rightarrow$ Entries ;

AXIOMS :
"axiomes concernant EntriesDirectory"
Ed1 : FileExists f = False $\Rightarrow$ EntriesDirectory e f not-defined ;
Ed2 : EntriesDirectory CreateInstance = *répertoire courant* ;
Ed3 : EntriesDirectory EntriesDirectory e f = f ;
"axiomes concernant IndexFile
(le prédicat GoodPath indique si un nom de fichier est acceptable)"
If1 : GoodPath f = False $\Rightarrow$ IndexFile e f not-defined ;
If2 : IndexFile CreateInstance = EmptyPath ;
If3 : IndexFile IndexFile e f = f ;

"axiomes concernant GenerateIndex"
Gi1 : GoodPath f = False ⇒ GenerateIndex e f not-defined ;
Gi2 : GenerateIndex CreateInstance not-defined ;
Gi3 : GenerateIndex e = GenerateIndex e IndexFile e ;
Gi4 : FileExists IndexFile GenerateIndex e = True ;
"axiomes concernant Entries"
En1 : Entries CreateInstance = EmptyEntries ;
WHERE :
e : Engine ;
f : Path ;

END ENGINE ;

D.1.2 Spécification de la coclasse Entries

SPEC : ENTRIES ;
USE : ENGINE, ENTRY, INTEGER, STRING, CLIPBOARD, PATH, FILE ;
SORT : Entries ;
"La spécification CLIPBOARD permet de spécifier le comportement du Presse-papiers.
La spécification FILE permet de simuler les fichiers textes. On utilisera notamment les opérations File p, où p est un Path, qui représente le fichier de nom p, et IsInFile f s qui renvoie True ssi la chaîne s figure dans le fichier f"

GENERATED BY :
"Ce générateur n'existe que dans la spécification formelle"
EmptyEntries : → Entries ;

OPERATIONS :
Count _ : Entries → Integer ;
Item _ _ : Entries Integer → Entry ;
Item _ _ : Entries String → Entry ;
Add _ _ : Entries String → Entries ;
Remove _ _ : Entries Integer → Entries ;
Remove _ _ : Entries String → Entries ;
RemoveAll _ : Entries → Entries ;
Paste _ _ : Entries Clipboard → Entries ;
Open _ _ : Entries Path → Entries ;
SaveAll _ : Entries → Entries ;
Retrieve _ _ : Entries String → Entries ;

PREDICATES :
"L'opération Contains n'apparaît que dans la spécification formelle
Elle renvoie True ssi la collection contient un élément portant le nom spécifié"
Contains _ _ : Entries String

AXIOMS :

"axiomes concernant Count"

Cn1 : Count EmptyEntries = 0 ;
Cn2 : Count Add e s = 1+ Count e ;
Cn3 : Count Remove e s = 1- Count e ;
Cn4 : Count Remove e n = 1- Count e ;
Cn5 : Count Paste e clp = 1+ Count e ;
Cn6 : Count Open e f = 1+ Count e ;
Cn8 : Count Retrieve e s = 1+ Count e ;

"axiomes concernant Item"

"$(e)_s$ représente l'élément de e qui porte le nom s"

It1 : (n < 1) or (n > Count e) $\Rightarrow$ Item e n not-defined ;
It2 : Count e = 1- n $\Rightarrow$ Item Add e s n = $(e)_s$;
It3 : Count e > 1- n $\Rightarrow$ Item Add e s n = Item e 1- n ;
It4 : Contains e s = False $\Rightarrow$ Item e s not-defined ;
It5 : Item Add e s s = $(e)_s$;
It6 : Equal s2 s = False $\Rightarrow$ Item Add e s2 s = Item e s ;
It7 : Item Remove e s2 s = Item e s ;

"Item Remove e n2 n n'est pas défini car l'ordre des éléments peut changer lors du retrait"

"Les axiomes pour Item relativement à Paste, Open et Retrieve ne sont pas directement formalisables"

"axiomes concernant Add"

Ad1 : Contains e s $\Rightarrow$ Add e s not-defined ;

"axiomes concernant Remove"

Rm1 : (n < 1) or (n > Count e) $\Rightarrow$ Remove e n not-defined ;
Rm2 : Contains e s = False $\Rightarrow$ Remove e s not-defined ;

"axiomes concernant RemoveAll"

Ra1 : RemoveAll e = EmptyEntries ;

"axiomes concernant Paste"

Ps1 : Equal DataType clp "Entry" = False $\Rightarrow$ Paste e clp not-defined ;

"axiomes concernant Open"

Op1 : FileExists f = False $\Rightarrow$ Open e f not-defined ;
Op2 : FileType f "peleas" = False $\Rightarrow$ Open e f not-defined ;

"axiomes concernant Retrieve"

Rt1 : IsInFile File IndexFile e s = False $\Rightarrow$ Retrieve e s not-defined ;

"aximes concernant Contains"

Co1 : Contains EmptyEntries s = False ;
Co2 : Contains Add e s s = True ;
Co3 : Equal s s2 = False $\Rightarrow$ Contains Add e s2 s = Contains e s ;

WHERE :

e : Entries ;
n : Integer ;
s, s2 : String ;
clp : Clipboard ;
f : Path ;

END ENTRIES ;

D.1.3 Spécification de la coclasse Entry

SPEC : ENTRY ;
USE : ENTRIES, DOMAINS, LIBRARIES, RULES, GUESSES,
STRING, CLIPBOARD, PATH, NODETYPE ;
SORT : Entry ;

GENERATED BY :
"Ce générateur n'existe que dans la spécification formelle"
Entry _ : → Entry ;

OPERATIONS :
Label _ : Entry → String ;
Label _ _ : Entry String → Entry ;
Domains _ : Entry → Domains ;
Copy _ _ : Entry Clipboard → Entry ;
Cut _ _ : Entry Clipboard → Entry ;
Libraries _ : Entry → Libraries ;
Rules _ : Entry → Rules ;
Guesses _ : Entry → Guesses ;
Path _ : Entry → String ;
Save _ _ : Entry Path → Entry
Evaluate _ : Entry → Entry ;
Type _ : Entry → NodeType ;

PREDICATES :
IsEvaluated _ : Entry ;

AXIOMS :
"axiomes concernant Label"
Lb1 : Label Entry s = s ;
Lb2 : Label Label e s = s ;
Lb3 : Label Copy e clp = Label e ;
Lb4 : Label Cut e clp = Label e ;
Lb5 : Label Save e p = Label e ;
Lb6 : Label Evaluate e = Label e ;
"axiomes concernant Domains"
Dm1 : Domains Entry s = EmptyDomains ;
Dm2 : Domains Label e s = Domains e ;
Dm3 : Domains Copy e clp = Domains e ;
Dm4 : Domains Cut e clp = Domains e ;
Dm5 : Domains Save e p = Domains e ;
Dm6 : Domains Evaluate e = Domains e ;
"les axiomes concernant Copy et Cut ne sont pas formalisables"

"axiomes concernant Libraries"
Li1 : Libraries Entry s = EmptyLibraries ;
Li2 : Libraries Label e s = Libraries e ;
Li3 : Libraries Copy e clp = Libraries e ;
Li4 : Libraries Cut e clp = Libraries ;
Li5 : Libraries Save e p = Libraries e ;
Li6 : Libraries Evaluate e = Libraries e ;
"axiomes concernant Rules"
Ru1 : Rules Entry s = EmptyRules ;
Ru2 : Rules Label e s = Rules e ;
Ru3 : Rules Copy e clp = Rules e ;
Ru4 : Rules Cut e clp = Rules ;
Ru5 : Rules Save e p = Rules e ;
Ru6 : Rules Evaluate e = Rules e ;
"axiomes concernant Guesses"
Gu1 : Guesses Entry s = EmptyGuesses ;
Gu2 : Guesses Label e s = Guesses e ;
Gu3 : Guesses Copy e clp = Guesses e ;
Gu4 : Guesses Cut e clp = Guesses ;
Gu5 : Guesses Save e p = Guesses e ;
Gu6 : Guesses Evaluate e = Guesses e ;
"axiomes concernant Path"
Pa1 : Path e = Label e ;
"les axiomes concernant Save et Evaluate ne sont pas formalisables"
"axiomes concernant Type"
Tp1 : Type e = pelENTRY ;
"axiomes concernant IsEvaluated"
Ie1 : IsEvaluated Entry s = False ;
Ie2 : IsEvaluated Label e s = IsEvaluated e ;
Ie3 : IsEvaluated Copy e clp = IsEvaluated e ;
Ie4 : IsEvaluated Cut e clp = IsEvaluated e ;
Ie5 : IsEvaluated Save e p = IsEvaluated e ;
Ie6 : IsEvaluated Evaluate e = True ;
WHERE :
e : Entry ;
s : String ;
clp : Clipboard ;
p : Path ;

END ENTRY ;

La spécification NODETYPE se contente de déclarer des valeurs codant les différents types de nœuds pouvant apparaître dans la structure descriptive d'une entrée lexicale, et sera mise en œuvre par le type `pelNodeType`.

SPEC : NODETYPE ;
SORT : Nodetype ;

GENERATED BY :
pelENTRY : → NodeType
pelDOMAIN : → NodeType
pelNOTION : → NodeType
pelCVIEW : → NodeType
pelFEATURE : → NodeType

END NODETYPE ;

D.1.4 Spécification de la coclasse Domains

SPEC : DOMAINS ;
USE : ENTRY, DOMAIN, INTEGER, STRING, CLIPBOARD ;
SORT : Domains ;

GENERATED BY :
"Ce générateur n'existe que dans la spécification formelle"
EmptyDomains : → Domains ;

OPERATIONS :
Count _ _ : Domains → Integer ;
Item _ _ : Domains Integer → Domain ;
Item _ _ : Domains String → Domain ;
Add _ _ : Domains String → Domains ;
Remove _ _ : Domains Integer → Domains ;
Remove _ _ : Domains String → Domains ;
RemoveAll _ : Domains → Domains ;
Paste _ _ : Domains Clipboard → Domains ;
Father _ : Domains → Entry ;

PREDICATES :
"N'apparaît que dans la spécification formelle"
Contains _ _ : Domains String ;

AXIOMS :
"axiomes concernant Count"
Cn1 : Count EmptyDomains = 0 ;
Cn2 : Count Add d s = 1+ Count d ;
Cn3 : Count Remove d n = 1- Count d ;
Cn4 : Count Remove d s = 1- Count d ;
Cn5 : Count Paste d clp = 1+ Count d ;
"axiomes concernant Item"
It1 : (n < 1) or (n > Count d) ⇒ Item d n not-defined ;
It2 : Count d = 1- n ⇒ Item Add d s n = $(d)_s$;
It3 : Count d > 1- n ⇒ Item Add d s n = Item d 1- n ;
It4 : Contains d s = False ⇒ Item d s not-defined ;
It5 : Item Add d s s = $(e)_s$;

It6 : Equal s2 s = False ⇒ Item Add d s2 s = Item d s ;
It7 : Item Remove d s2 s = Item d s ;
"axiomes concernant Add"
Ad1 : Contains d s = True ⇒ Add d s not-defined ;
"axiomes concernant Remove"
Rm1 : (n < 1) or (n > Count d) ⇒ Remove d n not-defined ;
Rm2 : Contains d s = False ⇒ Remove d s not-defined ;
"axiomes concernant RemoveAll"
Ra1 : RemoveAll d = EmptyDomains ;
"axiomes concernant Paste"
Ps1 : Equal DataType clp "Domain" = False ⇒ Paste d clp not-defined ;
"axiomes concernant Father"
Fa1 : Domains e = d ⇒ Father d = e ;
"axiomes concernant Contains"
Co1 : Contains EmptyDomains s = False ;
Co2 : Contains Add d s s = True ;
Co3 : Equal s s2 = False ⇒ Contains Add d s2 s = Contains d s ;

WHERE :

d : Domains ;
n : Integer ;
s, s2 : String ;
clp : Clipboard ;
e : Entry ;

END DOMAINS ;

D.1.5 Spécification de la coclasse Domain

SPEC : DOMAIN ;
USE : DOMAINS, NOTIONS, CONSTRAINTS, STRING,
ACTIVITY, CLIPBOARD, NODETYPE ;
SORT : Domain ;

GENERATED BY :

"Ce générateur n'existe que dans la spécification formelle"
Domain _ : String → Domain ;

OPERATIONS :

Label _ : Domain → String ;
Label _ _ : Domain String → Domain ;
Activity _ : Domain → Activity ;
Activity _ _ : Domain Activity → Domain ;
Notions _ : Domain → Notions ;
Copy _ _ : Domain Clipboard → Domain ;
Cut _ _ : Domain Clipboard → Domain ;
Constraints _ : Domain → Constraints ;
Path _ : Domain → String ;

Type _ : Domain → NodeType ;
Father _ : Domain → Domains ;

AXIOMS :

“axiomes concernant Label”
Lb1 : Label Domain s = s ;
Lb2 : Label Label d s = s ;
Lb3 : Label Activity d a = Label d ;
Lb4 : Label Copy d clp = Label d ;
Lb5 : Label Cut d clp = Label d ;
“axiomes concernant Activity”
Ac1 : Activity Domain s = pelUNKNOWN ;
Ac2 : Activity Label d s = Activity d ;
Ac3 : Activity Activity d a = a ;
Ac4 : Activity Copy d clp = Activity d ;
Ac5 : Activity Cut d clp = Activity d ;
“axiomes concernant Notions”
No1 : Notions Domain s = EmptyNotions ;
No2 : Notions Label d s = Notions d ;
No3 : Notions Activity d a = Notions d ;
No4 : Notions Copy d clp = Notions d ;
No5 : Notions Cut d clp = Notions d ;
“Les axiomes concernant Copy et Cut ne sont pas formalisables”
“axiomes concernant Constraints”
Co1 : Constraints Domain s = EmptyConstraints ;
Co2 : Constraints Label d s = Constraints d ;
Co3 : Constraints Activity d a = Constraints d ;
Co4 : Constraints Copy d clp = Constraints d ;
Co4 : Constraints Cut d clp = Constraints d ;
“axiomes concernant Path”
Pa1 : Path d = Path Father d + ’.’ + Label d ;
“axiomes concernant Type”
Ty1 : Type d = pelDOMAIN ;
“axiomes concernant Father”
Fa1 : Contains ds Label d = True ⇒ Father d = ds ;

WHERE :

d : Domain ;
ds : Domains ;
s : String ;
a : Activity ;
clp : Clipboard ;

END DOMAIN ;

La spécification ACTIVITY se contente de déclarer des valeurs codant les différents taux d’activité que peuvent prendres les nœuds dans la structure descriptive d’une entrée lexicale, et sera mise en œuvre par le type `pelActivity`.

SPEC : ACTIVITY ;
SORT : Activity ;

GENERATED BY :
pelUNKNOWN : → Activity
pelSALIENT : → Activity
pelVALID : → Activity
pelIGNORED : → Activity
pelNEGATED : → Activity

END ACTIVITY ;

D.1.6 Spécification du type pelConstraint

SPEC : CONSTRAINTTYPE ;
SORT : ConstraintType ;

GENERATED BY :
pelIMPLICATION : → ConstraintType ;
pelOPPOSITION : → ConstraintType ;
pelINCREASE : → ConstraintType ;
pelDECREASE : → ConstraintType ;

END CONSTRAINTTYPE ;

D.1.7 Spécification de la coclasse Constraint

SPEC : CONSTRAINT ;
USE : STRING, CONSTRAINTTYPE ;
SORT : Constraint ;

GENERATED BY :
"Ce générateur n'existe que dans la spécification formelle.
On ne peut pas formaliser ici les contraintes sur Add de
la spécification CONSTRAINTS."
Constraint _ _ _ : String ConstraintType String → Constraint ;

OPERATIONS :
InitialNode _ : Constraint → String ;
InitialNode _ _ : Constraint String → Constraint ;
Arrow _ : Constraint → ConstraintType ;
Arrow _ _ : Constraint ConstraintType → Constraint ;
FinalNode _ : Constraint → String ;

FinalNode _ _ : Constraint String → Constraint ;

AXIOMS :

"axiomes concernant InitialNode"
In1 : InitialNode Constraint s1 t s2 = s1 ;
In2 : InitialNode InitialNode c s = s ;
In3 : InitialNode Arrow c t = InitialNode c ;
In4 : InitialNode FinalNode c s = InitialNode c ;
"axiomes concernant Arrow"
Ar1 : Arrow Constraint s1 t s2 = t ;
Ar2 : Arrow InitialNode c s = Arrow c ;
Ar3 : Arrow Arrow c t = t ;
Ar4 : Arrow FinalNode c s = Arrow c ;
"axiomes concernant FinalNode"
Fn1 : FinalNode Constraint s1 t s2 = s2 ;
Fn2 : FinalNode InitialNode c s = FinalNode c ;
Fn3 : FinalNode Arrow c t = FinalNode c ;
Fn4 : FinalNode FinalNode c s = s ;

WHERE :

c : Constraint ;
s, s1, s2 : String ;
t : ConstraintType ;

END CONSTRAINT ;

D.1.8 Spécification de la coclasse Condition

SPEC : CONDITION ;
USE : STACK, BSTR, CONDTYPE, CLIPBOARD ;
SORT : Condition ;

GENERATED BY :

"Ces générateurs n'existent que dans la spécification formelle"
NewCond : → Condition ;
PredCond _ : String → Condition ;
AndCond _ _ : Condition Condition → Condition ;
OrCond _ _ : Condition Condition → Condition ;
NotCond _ : Condition Condition → Condition ;

OPERATIONS :

Text _ : Condition → String ;
Stack _ : Condition → Stack ;
ValidateTop _ : Condition → Condition ;
Type _ : Condition → ConditionType ;
"LComponent et RComponent sont des observateurs"
LComponent _ : Condition → Condition ;
RComponent _ : Condition → Condition ;

ValidateFromClipboard _ _ : Condition Clipboard → Condition ;
Unvalidate _ : Condition → Condition ;
MoveContentsToStack _ : Condition → Condition ;

AXIOMS :

"axiomes concernant Text"
Tx1 : Text NewCond = EmptyString ;
Tx2 : Text PredCond s = s ;
Tx3 : Text AndCond c1 c2 = "(" + Text c1 + ") AND ("
+ Text c2 + ")" ;
Tx4 : Text OrCond c1 c2 = "(" + Text c1 + ") OR ("
+ Text c2 + ")" ;
Tx5 : Text NotCond c = "NOT (" + Text c + ")" ;
"On axiomatise Text ValidateTop en passant plutôt par Type"
Tx6 : Type c = pelEMPTY ⇒ Text c = EmptyString ;
Tx7 : Type c = pelPREDICATE ⇒ Text c = Text LComponent c ;
Tx8 : Type c = pelAND ⇒ Text c = "(" + Text LComponentc
+ ") AND ("
+ Text RComponent c + ")" ;
Tx9 : Type c = pelOR ⇒ Text c = "(" + Text LComponentc
+ ") OR ("
+ Text RComponent c + ")" ;
Tx10 : Type c = pelNOT ⇒ Text c = "NOT ("
+ Text LComponent c + ")" ;
"Impossible de formaliser ici Text ValidateFromClipboard"
Tx11 : Text Unvalidate c = EmptyString ;
Tx12 : Text MoveContentsToStack c = EmptyString ;
"axiomes concernant Stack"
St1 : Stack NewCond = EmptyStack ;
St2 : Stack PredCond s = EmptyStack ;
St3 : Stack AndCond c1 c2 = EmptyStack ;
St4 : Stack OrCond c1 c2 = EmptyStack ;
St5 : Stack NotCond c = EmptyStack ;
St6 : Stack ValidateTop c = Stack c ;
St7 : Stack ValidateFromClipboard c clp = Stack c ;
St8 : Stack Unvalidate c = Stack c ;
St9 : Stack MoveContentsToStack c = Push LComponent c Stack c ;
"axiomes concernant Type"
Ty1 : Type NewCond = pelEMPTY ;
Ty2 : Type PredCond s = pelPREDICATE ;
Ty3 : Type AndCond c1 c2 = pelAND ;
Ty4 : Type OrCond c1 c2 = pelOR ;
Ty5 : Type NotCond c = pelNOT ;
Ty6 : Type ValidateTop c = Type Top Stack c ;
"Impossible de formaliser ici Type ValidateFromClipboard"
Ty7 : Type Unvalidate c = pelEMPTY ;
Ty8 : Type MoveContentsToStack c = pelEMPTY ;
"axiomes concernant LComponent"
Lc1 : LComponent NewCond = NewCond ;

```
    Lc2 : LComponent PredCond s = NewCond ;
    Lc3 : LComponent AndCond c1 c2 = c1 ;
    Lc4 : LComponent OrCond c1 c2 = c1 ;
    Lc5 : LComponent NotCond c = c ;
    Lc6 : LComponent ValidateTop c = LComponent Top Stack c ;
          "Impossible de formaliser ici
          LComponent ValidateFromClipboard"
    Lc7 : LComponent Unvalidate c = NewCond ;
    Lc8 : LComponent MoveContentsToStack c = NewCond ;
    "axiomes concernant RComponent"
    Rc1 : RComponent NewCond = NewCond ;
    Rc2 : RComponent PredCond s = NewCond ;
    Rc3 : RComponent AndCond c1 c2 = c2 ;
    Rc4 : RComponent OrCond c1 c2 = c2 ;
    Rc5 : RComponent NotCond c = NewCond ;
    Rc6 : RComponent ValidateTop c = RComponent Top Stack c ;
          "Impossible de formaliser ici
          RComponent ValidateFromClipboard"
    Rc7 : RComponent Unvalidate c = NewCond ;
    Rc8 : RComponent MoveContentsToStack c = NewCond ;

END CONDITION ;
```

La spécification CONDTYPE énumère les symboles qui codent les différents types de conditions possibles. Elle est mise en œuvre par le type `pelCondType`.

```
SPEC  : CONDTYPE ;
SORT  : ConditionType ;

GENERATED BY :
    pelEMPTY : → ConditionType ;
    pelPREDICATE : → ConditionType ;
    pelAND : → ConditionType ;
    pelOR : → ConditionType ;
    pelNOT : → ConditionType ;

END CONDTYPE ;
```

D.1.9 Spécification de la coclasse Stack

```
SPEC  : STACK ;
USE   : CONDITION, INTEGER, STRING, CONDTYPE, CLIPBOARD ;
SORT  : Stack ;

GENERATED BY :
```

"Le générateur n'existe que dans la spécification formelle"
EmptyStack : → Stack ;

OPERATIONS :

Size _ : Stack → Integer ;
Item _ _ : Stack Integer → Condition ;
Top _ : Stack → Condition ;
Push _ _ _ : Stack ConditionType String → Stack ;
Push _ _ : Stack ConditionType → Stack ;
InsertFromClipboard _ _ _ : Stack Clipboard Integer → Stack ;
Pop _ : Stack → Stack ;
PopAll _ : Stack → Stack ;
Remove _ _ : Stack Integer → Stack ;
UpInStack _ _ : Stack Integer → Stack ;
DownInStack _ _ : Stack Integer → Stack ;
Swap _ _ _ : Stack Integer Integer → Stack ;

AXIOMS :

"axiomes concernant Size"
Sz1 : Size EmptyStack = 0 ;
Sz2 : Size Push st t s = 1+ Size st ;
Sz3 : Size Push st t = 1+ Size st ;
Sz4 : Size InsertFromClipboard st clp n = 1+ Size st ;
Sz5 : Size Pop st = 1- Size st ;
Sz6 : Size PopAll st = 0 ;
Sz7 : Size Remove st n = 1- Size st ;
Sz8 : Size UpInStack st n = Size st ;
Sz9 : Size DownInStack st n = Size st ;
Sz10 : Size Swap st n1 n2 = Size st ;
"axiomes concernant Item"
It1 : (n < 1) or (n > Size st) ⇒ Item st n not-defined ;
It2 : n = 1 ⇒ Item Push st t s n = PredCond s ;
It3 : n < Size st ⇒ Item Push st t s n = Item st 1- n ;
It4 : (n = 1) and (Type Top st = pelAND) ⇒
Item Push st t n = AndCond Top Pop st Top Pop Pop st ;
It5 : (n = 1) and (Type Top st = pelOR) ⇒
Item Push st t n = OrCond Top Pop st Top Pop Pop st ;
It6 : (n = 1) and (Type Top st = pelNOT) ⇒
Item Push st t n = NotCond Top Pop st ;
It7 : (n = 1) and (Type Top st = pelEMPTY) ⇒
Item Push st t n not-defined ;
It8 : n < Size st ⇒ Item Push st t n = Item st 1- n ;
"Les axiomes concernant Item InsertFromClipboard ne sont pas formalisables ici"
It9 : 1 = Size st ⇒ Item Pop st n not-defined ;
It10 : Item Pop st n = Item st 1+ n ;
It11 : Item PopAll st n not-defined ;
It12 : n = Size st ⇒ Item Remove st n2 n not-defined ;
It13 : n < n2 ⇒ Item Remove st n2 n = Item st n ;

```
        It14 : n >= n2 ⇒ Item Remove st n2 n = Item st 1+ n ;
        It15 : (n < 1- n2) or (n > n2) ⇒ Item UpInStack st n2 n = Item st n ;
        It16 : n = 1- n2 ⇒ Item UpInStack st n2 n = Item st n2 ;
        It17 : Item UpInStack st n n = Item st 1- n2 ;
        It18 : (n < n2) or (n > 1+ n2) ⇒
               Item DownInStack st n2 n = Item st n ;
        It19 : n = 1+ n2 ⇒ Item DownInStack st n2 n = Item st n2 ;
        It20 : Item DownInStack st n n = Item st 1+ n ;
        It21 : (n ≠ n1) and (n ≠ n2) ⇒ Item Swap st n1 n2 n = Item st n ;
        It22 : Item Swap st n1 n2 n1 = Item st n2 ;
        It23 : Item Swap st n1 n2 n2 = Item st n1 ;
        "axiomes concernant Top"
        To1 : Top EmptyStack not-defined ;
        To2 : Top st = Item st 1 ;
        "axiomes concernant Push"
        Pu1 : t <> pelPREDICATES ⇒ Push st t s not-defined ;
        "axiomes concernant Pop"
        Po1 : Pop EmptyStack not-defined ;
        "axiomes concernant PopAll"
        Pa1 : PopAll st = EmptyStack ;
        "axiomes concernant Remove"
        Rm1 : (n < 1) or (n > Size st) ⇒ Remove st n not-defined ;
        "axiomes concernant UpInStack"
        Up1 : n > Size st ⇒ UpInStack st n not-defined ;
        Up2 : n = 1 ⇒ UpInStack st n = st ;
        "axiomes concernant DownInStack"
        Dn1 : n > Size st ⇒ DownInStack st n not-defined ;
        Dn2 : n = Size st ⇒ DownInStack st n = st ;
        "axiomes concernant Swap"
        Sw1 : ((n1 < 1) or (n1 > Size st) or (n2 < 1) or (n2 > Size st)) ⇒
              Swap st n1 n2 not-defined ;
WHERE :
        st : Stack ;
        n, n1, n2 : Integer ;
        s : String ;
        t : ConditionType ;
        clp : Clipboard ;

END STACK ;
```

D.2 Spécifications des coclasses d'AMORPH

D.2.1 Spécification de la coclasse AnaLex

```
SPEC  : ANALEX ;
USE   : STRING, BOOL, INTEGER, FLEXEDFORMS ;
SORT  : AnaLex ;
```

GENERATED BY :
CreateInstance : -> AnaLex ;

OPERATIONS :
"consulter ou modifier le chemin de la base de données"
LexicalDB _ : AnaLex → String ;
LexicalDB _ _ : AnaLex String → AnaLex ;
"charger la base de données"
Load _ : AnaLex → AnaLex ;
Load _ _ : AnaLex String → AnaLex ;
"indique si la base de données est chargée"
Loaded _ : AnaLex → Bool ;
"effectue l'analyse morphologique d'un mot"
Analyze _ _ : AnaLex String → AnaLex ;
"indique si la dernière opération d'analyse n'a pas été vaine"
Known _ : AnaLex → Bool ;
"renvoie la collection des analyses possibles"
FlexedForms _ : AnaLex → FlexedForms ;

AXIOMS :
"axiomes concernant LexicalDB"
Db1 : LexicalDB CreateInstance = "pitrat.mdb" ;
Db2 : LexicalDB LexicalDB a s = s ;
Db3 : LexicalDB Load a = LexicalDB a ;
Db4 : LexicalDB Load a s = s ;
Db5 : LexicalDB Analyze a s = LexicalDB a ;
"axiomes concernant Loaded"
Ld1 : Loaded CreateInstance = False ;
Ld2 : Loaded LexicalDB a s = False ;
Ld3 : Loaded Load a = True ;
Ld4 : Loaded Load a s = True ;
Ld5 : Loaded Analyze a s = Loaded a ;
"axiomes concernant Analyze"
An1 : Loaded a = False ⇒ Analyze a s not-defined ;
"axiomes concernant Known"
Kn1 : Known CreateInstance = False ;
Kn2 : Known LexicalDB a s = False ;
Kn3 : Known Load a = False ;
Kn4 : Known Load a s = False ;
Kn5 : Known Analyze a s = (Count FlexedForms a > 0) ;
WHERE :
a : AnaLex ;
s : String ;

END ANALEX ;

D.2.2 Spécification de la coclasse FlexedForms

SPEC : FLEXEDFORMS ;
USE : INTEGER, FLEXEDFORM ;
SORT : FlexedForm ;

GENERATED BY :
"il n'y a pas de générateur explicite"
"en fait FlexedForms représente l'union des tables engendrées par"
"les dernières requêtes Sigma émises par l'instance de AnaLex"
"à qui appartient cette collection"

OPERATIONS :
Count _ : FlexedForms $\rightarrow$ Integer ;
Item _ _ : FlexedForms Integer $\rightarrow$ FlexedForm ;

AXIOMS :
Cn1 : Count f = $\left| \bigcup_{\rho+\delta} \Sigma(\rho,\delta) \right|$
It1 : Count f = 0 => Item f n not-defined ;
It2 : n <= 0 or n > Count f $\Rightarrow$ Item f n not-defined ;
It3 : Item f n = $\left(\bigcup_{\rho+\delta} \Sigma(\rho,\delta) \right)_n$
WHERE :
f : FlexedForms ;
n : Integer ;

END FLEXEDFORMS ;

D.2.3 Spécification de la coclasse FlexedForm

SPEC : FLEXEDFORM ;
USE : STRING, INTEGER ;
SORT : FlexedForm ;

GENERATED BY :
"pas de générateur explicite"
"une instance de FlexedForm est un n-uplet d'une table"
"FlexedForms à qui appartient cet élément"

OPERATIONS :
CanonicalForm _ : FlexedForm $\rightarrow$ String ;
Group _ : FlexedForm $\rightarrow$ String ;
Tense _ : FlexedForm $\rightarrow$ String ;
Person _ : FlexedForm $\rightarrow$ Integer ;

AXIOMS :
Cf1 : CanonicalForm f = $\Pi_{\text{FORME-CAN}}$(f)
Gr1 : Group f = $\Pi_{\text{NOM-GROUPE}}$(f)
Ts1 : Tense f = $\Pi_{\text{NOM-TEMPS}}$(f)
Pr1 : Person f = Π_{PERS}(f)
WHERE :
f : FlexedForm ;

END FLEXEDFORM ;

Annexe E

Base de données morphologique

Les données contenues dans la base de l'analyseur morphologique appartiennent au domaine public et sont disponibles, entre autres, au LIMSI sur simple demande (on peut, à cet effet, consulter la page Web du LIMSI à l'adresse `http://www.limsi.fr`).

Cependant, pour des raisons d'optimisation du stockage de ces données (qui sont fournies sous forme de fichiers ASCII formatés) et d'efficacité du traitement, nous avons choisi de les intégrer au sein d'un SGBD/R. Celui qui a retenu notre attention est le SGBD/R de Microsoft : Access. En effet, il fait partie de la collection de logiciels bureautiques Office 97 et est le plus souvent intensivement utilisé dans l'environnement de travail des traducteurs. Ainsi, partant du principe que l'application Access est toujours lancée, au moins en tâche de fond, le fait d'y recourir pour gérer nos données, ne grève pas les ressources du système d'exploitation. Même en dehors de cet aspect bureautique, le moteur d'Access crée une empreinte mémoire d'environ 1,5 Mo, et gère les données par échange de pages de 2 Ko. C'est pourquoi, comme nous l'avons indiqué dans les données techniques ci-dessus, l'empreinte maximale en mémoire de l'analyseur AMORPH (en incluant le moteur Access et la base de données) ne dépasse pas 1,8 Mo si Access n'était pas précédemment chargé, et 300 Ko environ sinon.

E.1 Schémas de la base de données

Les données d'un analyseur de type Pitrat sont organisées en trois tables principales :

1. *La table des mots* contient, pour chaque forme canonique du lexique, des références aux conjugaisons de la forme canonique, ainsi que les racines utilisées par ces conjugaisons. On remarquera qu'une forme canonique peut avoir un nombre arbitraire de conjugaisons qui, à leur tour, utilisent chacune un nombre arbitraire de racines.

2. *La table des conjugaisons* décrit, pour chaque conjugaison connue, les groupes possibles pour cette conjugaison (par exemple "indicatif" pour une conjugaison de verbe), les jeux de terminaisons possibles pour un groupe (comme le jeu "indicatif présent des verbes du 1er groupe"), et pour chaque jeu, le numéro de la racine (contenue dans la table des mots) correspondant à chaque déclinaison ou personne figurant dans ce jeu. Comme pour la table des mots, une conjugaison peut avoir un nombre arbitraire de groupes, incluant un nombre arbitraire de jeux de terminaisons, chacune utilisant un nombre arbitraire de racines.

3. *La table des jeux de terminaisons* liste, pour chaque jeu de terminaison, l'ensemble des désinences correspondant aux déclinaisons de ce jeu. Il y a, bien sûr, un nombre arbitraire de désinences pour chaque jeu.

Le modèle relationnel ne permet pas de gérer des attributs dont les valeurs sont des listes arbitrairement longues d'éléments. D'ordinaire, l'usage est de remplacer un tel attribut par un identificateur permettant de retrouver la liste des éléments dans une autre table. Cependant, cette méthode implique d'engendrer un identificateur unique pour chaque liste différente. Or, pour les données dont nous disposons ici, la combinatoire *racines* $\times$ *conjugaisons* $\times$ *groupes* $\times$ *jeux de terminaisons* $\times$ *désinences* produit un nombre extrêmement important de listes différentes les unes des autres. Par conséquent, l'approche habituelle de fractionnement des tables serait beaucoup trop coûteuse en espace, à cause des identificateurs de listes introduits, et en temps car elle nous obligerait à faire de nombreuses jointures.

Nous avons donc choisi une autre approche, qui consiste à numéroter chaque variante d'une liste de valeurs dans une table, et à définir la clé primaire de cette table comme étant l'union de la clé primaire originale et des attributs contenant les numéros de variantes. Cette méthode a l'avantage sur la précédente d'utiliser comme identificateurs des entiers courts (donc très facilement optimisables par le SGBD/R) dont les valeurs ne sont pas uniques (soit un gain en espace), et d'éviter les jointures avec les tables de listes (soit un gain en temps). Cette méthode présente en outre l'avantage de conserver la classe de normalité des schémas munis de leurs dépendances fonctionnelles. Dans notre cas, toutes nos tables seront ainsi en Forme Normale de Boyce-Codd.

En conséquence, les schémas des trois tables citées plus hauts sont les suivants.

E.1.1 Schéma de la table des mots

Dans les fichiers formatés que nous avions utilisés au départ, une entrée correspondant à une forme canonique est de la forme :

```
Forme canonique
      Conjugaison-1 Racine-1-1 Racine-1-2 ... Racine-1-n
      ...
      Conjugaison-p Racine-p-1 Racine-p-2 ... Racine-p-m
```

Cette entrée sera représentée dans notre table des mots par plusieurs n-uplets, un pour chaque combinaison *conjugaison* $\times$ *racine*.

Le schéma de la table des mots sera donc :

$$\textit{MOTS} = (\text{FORM-CAN}, \text{NUM-CONJ}, \text{CONJ}, \text{NUM-RAC}, \text{RAC})$$

Du fait de cette représentation, notre table des mots contiendra environ 40.000 n-uplets.

E.1.2 Schéma de la table des conjugaisons

Une entrée correspondant à la définition d'une conjugaison dans les fichiers formatés est de la forme :

```
Conjugaison
      Groupe-1
```

DICTIONNAIRE DES DONNEES				
Attribut	**Désignation**	**Domaine**	**Longueur**	**Remarques**
FORME-CAN	Forme canonique	Alphanumérique	40 car.	Entrée du fichier formaté
NUM-CONJ	Numéro de conjugaison	Numérique	8 bits	Rang de cette conjugaison dans la liste
CONJ	Identificateur de conjugaison	Alphanumérique	10 car.	Référence à la table CONJUGAISONS
NUM-RAC	Numéro de racine	Numérique	8 bits	Rang de cette racine dans la liste
RAC	Racine	Alphanumérique	20 car.	

FIG. E.1: Dictionnaire de la table *MOTS*

```
        Terminaison-1-1 Num-racine-1-1-1 ... Num-racine-1-1-n
        ...
        Terminaison-1-m Num-racine-1-m-1 ... Num-racine-1-m-p
...
Groupe-k
        Terminaison-k-1 Num-racine-k-1-1 ... Num-racine-k-1-q
        ...
        Terminaison-k-r Num-racine-k-r-1 ... Num-racine-k-r-s
```

De façon similaire à ce que nous avons fait pour la table des mots, cette entrée sera représentée par un n-uplet pour chaque combinaison *groupe* $\times$ *terminaison* $\times$ *numéro de racine*. Et le schéma de la table des conjugaisons est :

CONJUGAISONS = (CONJ, NUM-GROUPE, GROUPE, NUM-TERM,
TERM, PERS, NUM-RAC)

L'attribut PERS est le numéro du numéro de racine, ce qui correspond à la "personne" si on considère la conjugaison d'un verbe.

Du fait de cette représentation, notre table des conjugaisons contient environ 2.900 n-uplets.

E.1.3 Schéma de la table des terminaisons

Une entrée correspondant à la définition d'un jeu de terminaisons dans les fichiers formatés est de la forme :

```
Terminaison Désinence-1 ... Désinence-n
```

Cette entrée sera donc représentée par un n-uplet pour chaque combinaison *terminaison* $\times$ *désinence*. Et le schéma de la table des jeux de terminaisons est :

TERMINAISONS = (TERM, NUM-DES, DES)

Cela nous amène donc à une table contenant 510 n-uplets.

DICTIONNAIRE DES DONNEES				
Attribut	**Désignation**	**Domaine**	**Longueur**	**Remarques**
CONJ	Identificateur de conjugaison	Alphanumérique	10 car.	Identifie de façon unique tous les éléments d'une conjugaison
NUM-GROUPE	Numéro de groupe	Numérique	8 bits	Range de ce groupe dans la liste
GROUPE	Identificateur de groupe	Alphanumérique	10 car.	Référence à la table GROUPES
NUM-TERM	Numéro du jeu de terminaisons	Numérique	8 bits	Rang de ce jeu dans la liste
TERM	Identificateur de jeu de terminaison	Alphanumérique	10 car.	Référence aux tables TERMINAISON et TEMPS
PERS	Numéro du numéro de racine utilisée	Numérique	8 bits	Rang de ce numéro de racine dans la liste
NUM-RAC	Numéro de la racine utilisée	Numérique	8 bits	Numéro de racine figurant dans la table MOTS

FIG. E.2: Dictionnaire de la table *CONJUGAISONS*

DICTIONNAIRE DES DONNEES				
Attribut	**Désignation**	**Domaine**	**Longueur**	**Remarques**
TERM	Identificateur de jeu de terminaisons	Alphanumérique	10 car.	Identifie de façon unique toutes les déclinaisons d'un jeu de terminaisons
NUM-DES	Numéro de désinence	Numérique	8 bits	Range de cette désinence dans la liste
DES	Désinence	Alphanumérique	20 car.	

FIG. E.3: Dictionnaire de la table *TERMINAISONS*

E.1.4 Schémas supplémentaires

Les données contenues dans les fichiers formatés utilisés contiennent beaucoup d'information sous forme non directement lisible, notamment les noms de groupes et les noms de jeux de terminaisons. Nous avons donc rajouté deux tables supplémentaires permettant de connaître la forme lisible d'un groupe, et celle d'un jeu de terminaisons à partir de leurs identificateurs. Par exemple, dans nos données, l'identificateur de groupe *ip* correspond à "indicatif" et *ip6* à "présent".

Ces deux tables supplémentaires ont pour schémas :

GROUPES = (GROUPE, NOM-GROUPE)

TEMPS = (TERM, NOM-TEMPS)

DICTIONNAIRE DES DONNEES				
Attribut	**Désignation**	**Domaine**	**Longueur**	**Remarques**
GROUPE	Identificateur de groupe	Alphanumérique	20 car.	Identifie un groupe de façon unique
NOM-GROUPE	Nom lisible du groupe	Alphanumérique	20 car.	
TERM	Identificateur de jeu de terminaisons	Alphanumérique	10 car.	Identifie un jeu de terminaison de façon unique
NOM-TEMPS	Nom lisible du jeu de terminaisons	Alphanumérique	20 car.	

FIG. E.4: Dictionnaires des tables *GROUPES* et *TEMPS*

Par analogie avec la conjugaison des verbes, nous avons nommé it TEMPS les jeux de terminaisons, même s'ils peuvent concerner les déclinaisons des noms communs, adjectifs, etc.

E.2 Algorithme d'analyse morphologique

L'algorithme qui réalise l'analyse morphologique d'une forme fléchie à partir de ces données, tel que présenté par Sabah [106] est le suivant :

1. Découper la forme fléchie en un radical ρ et une désinence δ. Ce découpage est arbitraire.
2. Chercher l'ensemble des jeux de terminaisons contenant cette désinence $R_T(\delta) = \{T_i\}$.
3. Chercher l'ensemble des formes canoniques candidates pour cette forme fléchie, c'est-à-dire l'ensemble des "mots" qui utilisent ρ comme racine : $R_C(\rho) = \{(FC_i, C_i, NR_i)\}$ où FC est une forme canonique, C est une conjugaison et NR est le numéro que porte la racine ρ pour cette conjugaison de cette forme canonique.
4. On vérifie ensuite qu'il y a correspondance, pour au moins un triplet (FC_i, C_i, NR_i) avec la conjugaison C_i qui doit effectivement utiliser un des jeux de terminaisons contenus dans $R_T(\delta)$.

5. S'il y a correspondance, on récupère la forme canonique, le groupe, le jeu de terminaisons ainsi que le numéro de désinence (i.e. la "personne"). Puis on met en correspondance le groupe et le jeu de terminaisons avec les deux tables *GROUPES* et *TEMPS* afin d'obtenir leurs noms de façon lisible.

6. L'ensemble des étapes 1 à 5 doit être effectué pour toutes les décompositions $\rho + \delta$ possibles de la forme fléchie. Si on ne trouve aucune correspondance pour aucune décomposition, c'est que la forme fléchie est inconnue ou mal orthographiée.

E.3 Requêtes d'analyse morphologique

Les étapes 1 à 5 de l'algorithme présenté ci-dessus peuvent être écrites sous forme de requêtes dans la base de données morphologiques. Nous les présentons en algèbre relationnelle.

Pour une décomposition $\rho + \delta$ donnée, on peut effectuer les étapes 1 à 5 de l'algorithme en une seule requête paramétrée dépendant de ρ et de δ :

$$S(\rho, \delta) = \Pi_{\substack{\text{FORME-CAN},\\ \text{NOM-GROUPE},\\ \text{NOM-TEMPS},\\ \text{PERS}}} \left(\sigma_{\substack{\text{RAC} = \rho,\\ \text{DES} = \delta}} (\mathcal{J}) \right)$$

où la vue $\mathcal{J}$ est définie par :

$$\mathcal{J} = \textit{MOTS} \underset{\substack{\text{CONJ},\\ \text{NUM-RAC}}}{\bowtie} \textit{CONJUGAISONS} \underset{\text{TERM}}{\bowtie} \textit{TERMINAISONS} \ldots \ldots \underset{\text{GROUPE}}{\bowtie} \textit{GROUPES} \underset{\text{TERM}}{\bowtie} \textit{TEMPS}$$

Bien entendu, on ne peut se satisfaire d'une telle requête comportant quatre jointures enchaînées, ce qui engendre, dans notre cas, un produit cartésien total d'environ $6,74 \times 10^{13}$ n-uplets. Ceci est absolument inacceptable du point de vue de l'occupation en mémoire, ainsi que du temps de traitement.

Nous allons donc utiliser des requêtes optimisées en :

1. enfonçant les sélections le plus profond possible à l'intérieur de la formule ;

2. ajoutant des projections aussitôt que possible afin de ne conserver que les attributs absolument nécessaires pour réaliser les opérations algébriques situées plus haut dans la formule ;

3. découpant cette requête en sous-requêtes de manière à limiter le nombre de jointures enchaînées.

Nous définissons ainsi les cinq requêtes suivantes, correspondant en fait aux cinq étapes de l'algorithme donné dans la section précédente (le paramètre ρ y est nommé Radical, et le paramètre δ est nommé Desinence) :

— $R_T(\delta) = \Pi_{\text{TERM}}\big(\sigma_{\text{DES}=\delta}(\textit{TERMINAISONS})\big)$, ce qui donne en SQL :

```
PARAMETERS Desinence Text;
SELECT DISTINCT Terminaisons.[Term]
FROM Terminaisons
WHERE (((Terminaisons.Des)=[Desinence]));
```

— $R_C(\rho) = \Pi_{\text{FORME-CAN,CONJ,NUM-RAC}}\big(\sigma_{\text{RAC}=\rho}(\textit{MOTS})\big)$, soit :

```
PARAMETERS Radical Text;
SELECT DISTINCT Mots.[Forme-Can], Mots.[Conj], Mots.[Num-Rac]
FROM Mots
WHERE (((Mots.Racine)=[Radical]));
```

— $\Sigma_0(\rho) = \Pi_{\substack{\text{FORME-CAN,}\\ \text{GROUPE,}\\ \text{TERM,}\\ \text{PERS}}} \left(R_C(\rho) \underset{\substack{\text{CONJ,}\\ \text{NUM-RAC}}}{\bowtie} \textit{CONJUGAISONS} \right)$, soit en SQL :

```
SELECT DISTINCT RC.[Forme-Can], Conjugaisons.[Groupe],
                Conjugaisons.[Term], Conjugaisons.[Pers]
FROM RC INNER JOIN Conjugaisons
ON (RC.[Num-Rac] = Conjugaisons.[Num-Rac])
AND (RC.[Conj] = Conjugaisons.[Conj]);
```

— $\Sigma_1(\rho,\delta) = \Pi_{\substack{\text{FORME-CAN,}\\ \text{GROUPE,}\\ \text{TERM,}\\ \text{PERS}}} \left(\Sigma_0(\rho) \underset{\text{TERM}}{\bowtie} R_T(\delta) \right)$, soit

```
SELECT DISTINCT Sigma0.[Forme-Can], Sigma0.[Groupe],
                Sigma0.[Term], Sigma0.[Pers]
FROM Sigma0 INNER JOIN RT
ON Sigma0.[Term] = RT.[Term];
```

— $\Sigma(\rho,\delta) = \Pi_{\substack{\text{FORME-CAN,}\\ \text{NOM-GROUPE,}\\ \text{NOM-TEMPS,}\\ \text{PERS}}} \left(\textit{GROUPES} \underset{\text{GROUPE}}{\bowtie} \left(\Sigma_1(\rho,\delta) \underset{\text{TERM}}{\bowtie} \textit{TEMPS} \right) \right)$, soit

```
SELECT Sigma1.[Forme-Can], Groupes.[Nom-Groupe],
       Temps.[Nom-Temps], Sigma1.[Pers]
FROM (Groupes INNER JOIN Sigma1 ON Groupes.Groupe = Sigma1.Groupe)
INNER JOIN Temps
ON Sigma1.Term = Temps.Term;
```

C'est donc la requête $\Sigma(\rho,\delta)$ qui va nous fournir le résultat de l'analyse morphologique correspondant à la décomposition $\rho + \delta$.

Annexe F

Représentation hypergraphique des entrées lexicales

F.1 Motivation de l'équivalence graphe / hypergraphe

Il y a deux raisons pour nous de chercher à établir une équivalence formelle entre la représentation d'un potentiel sémantique PELEAS sous forme de graphe, comme nous l'avons présenté jusqu'à présent, et une représentation sous forme d'hypergraphe. Ces deux raisons se fondent sur l'identité structurelle existant entre deux niveaux du graphe, et ce quels que soient les niveaux.

Tout d'abord, d'un point de vue strictement linguistique, il peut être utile au terminologue qui maintient la base de potentiels sémantiques d'avoir accès à deux représentations du même objet linguistique en fonction de ses objectifs du moment :

— soit avoir une vision d'ensemble du potentiel sémantique lui permettant d'observer les pôles sémantiques qui constituent le squelette du potentiel sémantique, c'est-à-dire les chemins du graphe allant de la racine entrée lexicale jusqu'aux feuilles. La représentation par graphe est tout indiquée pour cet objectif puisqu'elle constitue une mise à plat du potentiel sémantique sous forme de chemins et de relations entre ces chemins ;

— soit avoir une vision locale d'un sommet particulier du graphe et de ses fils afin de pouvoir établir plus finement les relations existant entre ces fils. La structure {sommet, fils de ce sommet} doit donc pouvoir être constituée en tant qu'unité observable autonome, et une représentation par hypergraphe est un bon candidat, comme nous allons le montrer.

Le processus d'expertise terminologique requis pour la construction et la maintenance de la base de potentiels sémantiques nécessite donc un va-et-vient constant entre ces deux visions, ce qui nous a conduit à rechercher un moyen à la fois théorique et pratique pour passer aisément de la vision globale à la vision locale, et vice versa.

D'autre part, il existe une raison encore plus pratique à l'emploi d'une représentation par hypergraphe : du point de vue de la mise en œuvre informatique, au niveau du codage, il nous fallait choisir une représentation de la structure de graphe qui soit à la fois efficace et suffisamment puissante. Et c'est en observant l'homogénéité de la structure de deux niveaux consécutifs du graphe que nous avons choisi de tirer parti de l'orientation objets du langage C++ qui induit une représentation naturelle de notre structure de graphe par un hypergraphe. En effet, si les sommets de chaque niveau d'un potentiel sémantique

présentent un comportement légèrement différent de ceux des autres niveaux en ce qui concerne la propagation de l'activité, tous les sommets de tous les niveaux entretiennent des relations identiques avec leur père, leurs frères et leurs fils. Nous avons donc choisi de faire dériver tous les sommets d'une même classe mère qui contient une information minimale : le sommet lui-même (c'est-à-dire son étiquette et son activité) et ses fils (ou des pointeurs vers ses fils) avec les relations qu'ils entretiennent entre eux. On retrouve donc l'unité d'observation terminologique locale évoquée plus haut. De plus, on voit qu'un sommet ainsi représenté est intuitivement un graphe en soi, puisque l'ensemble des fils de ce sommet et leurs relations (qui sont contenus dans le sommet) forment un graphe *stricto sensu*.

Ainsi, à la fois les contingences du codage et les besoins de l'expertise terminologique nous conduisent à utiliser une représentation du graphe du potentiel sémantique par un hypergraphe. Ceci soulève des questions théoriques puisqu'il y a là une supposition d'équivalence à la fois en expressivité et en puissance entre une structure homogène "plate" (le graphe) et une structure hiérarchique pouvant comporter jusqu'à cinq niveaux d'emboîtement (l'hypergaphe). Donc, si nous parvenons à établir une équivalence stricte entre le type de graphe que nous utilisons et un type d'hypergraphe, nous aurons mis en évidence l'existence d'une structure de données "plate" mais implicitement hiérarchique, et surtout qui permet l'emploi d'une notion de contexte local sans utiliser de structure à base de boîtes ou de *vista*.

F.2 Définition d'une bijection

Pour établir cette équivalence, nous allons chercher à montrer qu'il existe une bijection entre les graphes pyramidaux et un type d'hypergraphe. Pour prouver l'existence de cette bijection, nous allons la construire explicitement. Pour cela, nous allons exhiber une série de foncteurs entre les sommets d'un graphe pyramidal ainsi que leurs fils, et une classe d'hypergraphes que nous allons définir. Comme nous allons le montrer, la méthode de construction des foncteurs est semblable pour tous les types de sommets à partir des feuilles ; par induction structurelle sur les différents types de sommets ordonnés, il est alors trivial d'en déduire une bijection entre cette classe d'hypergraphes et la classe des graphes pyramidaux, quel que soit le nombre de niveaux qu'ils comportent.

Nous allons tout d'abord définir la classe d'hypergraphes que nous allons mettre en bijection avec les graphes pyramidaux.

Définition 10 *Une* pyramide, *paramétrée par les ensembles Σ, V et $R(V)$, est définie comme un triplet $\langle S_{\Sigma,V,R(V)}, A_{R(V)}, E_{\Sigma,V} \rangle$ où $S_{\Sigma,V,R(V)}$ est un ensemble fini de sommets, $A_{R(V)}$ est un ensemble fini d'arcs étiquetés par des éléments de $R(V)$, et $E_{\Sigma,V}$ est un ensemble fini de* propriétés *qui sont des paires $<$ étiquette de la propriété, valeur de la propriété $>$.*

Cette structure possède les propriétés suivantes :

- *tous les éléments de $S_{\Sigma,V,R(V)}$ sont eux-mêmes des pyramides paramétrées par Σ, V et $R(V)$;*
- *l'ensemble $E_{\Sigma,V}$ contient au moins les deux propriétés $<$ étiquette$, \sigma >$ ($\sigma \in \Sigma$), ainsi que $<$ valeur$, v >$ ($v \in V$).*

F.2.1 Construction de la bijection pour les graphes pyramidaux de profondeur 1

Soit $G_{\Sigma,V,R(V)}$ un graphe pyramidal de profondeur 1, c'est-à-dire qu'il comporte un unique sommet s qui est à la fois racine et feuille, et qu'il ne contient pas d'arc.

A $G_{\Sigma,V,R(V)}$, on peut associer une unique pyramide

$$P_{\Sigma,V,R(V)} = \langle \emptyset, \emptyset, \{< \text{étiquette}, \text{etiq}(s) >, < \text{valeur}, \text{val}(s) >\}\rangle$$

qui ne comporte ni sommet, ni arc, mais reprend les deux propriétés du graphe pyramidal $G_{\Sigma,V,R(V)}$ (i.e. la seule information qu'il contient).

Trivialement, la réciproque est vraie pour toute pyramide de ce genre, puisqu'un graphe pyramidal de profondeur 1 se résume à un unique sommet repéré par son étiquette et sa valeur. Il est donc possible de définir une bijection F pour laquelle l'unique image de $G_{\Sigma,V,R(V)}$ est $P_{\Sigma,V,R(V)}$ et l'unique antécédent de $P_{\Sigma,V,R(V)}$ est $G_{\Sigma,V,R(V)}$.

Puisque nous n'avons restreint ni $G_{\Sigma,V,R(V)}$, ni $P_{\Sigma,V,R(V)}$, F est une bijection entre l'ensemble des graphes pyramidaux de profondeur 1 et un sous-ensemble des pyramides que nous appellerons $\mathcal{P}^1_{\Sigma,V,R(V)}$.

F.2.2 Extension de la bijection de la profondeur n à la profondeur $n+1$

Supposons que l'on dispose de la bijection F étendue aux graphes pyramidaux de profondeur n, mis en bijection avec un sous-ensemble de pyramides $\mathcal{P}^n_{\Sigma,V,R(V)}$. Nous allons montrer qu'il est possible d'étendre F aux graphes pyramidaux de profondeur $n+1$, en définissant un nouveau sous-ensemble de pyramides $\mathcal{P}^{n+1}_{\Sigma,V,R(V)}$.

Soit $G_{\Sigma,V,R(V)} = \left\langle \mathcal{A}_{\Sigma,V,R(V)}, A'_{R(V)}, A''_{R(V)} \right\rangle$ un graphe pyramidal de profondeur $n+1$ (où $\mathcal{A}_{\Sigma,V,R(V)} = \left\langle S_{\Sigma,V}, A_{R(V)} \right\rangle$ est l'arbre n-aire sous-jacent du graphe pyramidal).

On va définir une injection K (qui est en fait un foncteur) de l'ensemble des $G_{\Sigma,V,R(V)}$ possibles dans l'ensemble des pyramides. Ce foncteur associe à $G_{\Sigma,V,R(V)}$ une unique pyramide définie par :

$$K(G_{\Sigma,V,R(V)}) = \Big\langle K_0(S_{\Sigma,V}), K_1(A_{R(V)} \cup A'_{R(V)} \cup A_{R(V)}), \\ \big\{< \text{étiquette}, \text{etiq}(\text{Rac}(G_{\Sigma,V,R(V)})) >, < \text{valeur}, \text{val}(\text{Rac}(G_{\Sigma,V,R(V)})) >\big\}\Big\rangle$$

On définit les applications K_0 et K_1 comme suit :

— Soient $s_1, \ldots, s_p$ les sommets fils de la racine du graphe pyramidal $G_{\Sigma,V,R(V)}$, et soient $G^{s_1}_{\Sigma,V,R(V)}$, $\ldots$, $G^{s_p}_{\Sigma,V,R(V)}$ les sous-graphes de $G_{\Sigma,V,R(V)}$ qui ont pour racine ces sommets.
Alors $K_0(S_{\Sigma,V}) = \{F(G^{s_1}_{\Sigma,V,R(V)}), \ldots, F(G^{s_p}_{\Sigma,V,R(V)})\}$.

— $K_1(A_{R(V)} \cup A'_{R(V)} \cup A_{R(V)}) = \{K_1(s_i \xrightarrow{f} s_j), s_i \xrightarrow{f} s_j \in A_{R(V)} \cup A'_{R(V)} \cup A_{R(V)}\}$.

Les $K_1(s_i \xrightarrow{f} s_j)$ sont les transpositions, dans l'ensemble des pyramides, des arcs existant entre les sommets du graphe pyramidal $G_{\Sigma,V,R(V)}$. Pour comprendre quelles relations sont établies par le foncteur K, nous allons donner des définitions plus formelles des éléments manipulés, et montrer que K est en fait un homéomorphisme.

Tout d'abord, notons le rôle prépondérant de l'étiquette et de la valeur d'un sommet, qu'il appartienne à un graphe pyramidal ou à une pyramide. Pour uniformiser les notations, on considèrera les fonctions suivantes, qui donnent l'étiquette et la valeur respectivement d'un graphe pyramidal (pour un graphe pyramidal G, $\text{etiq}_g(G) = \text{etiq}(\text{Rac}(G))$ et $\text{val}_g = \text{val}(\text{Rac}(G))$), et d'une pyramide ($\text{etiq}_p$ et val_p).

Nous pouvons maintenant signaler que nous choisissons de construire le foncteur K de telle façon que $K(\mathrm{val}_g) = \mathrm{val}_p$.

Observons que $R(V)$ est un ensemble fini, dénombrable de q fonctions de V dans V. Nous allons noter $\gamma_1, \ldots, \gamma_q$ ces fonctions. Elles induisent autant de relations $\top_1, \ldots, \top_q$ sur les sous-graphes du graphe pyramidal. Ainsi, si x et y sont deux sous-graphes $G_{\Sigma,V,R(V)}$,

$$x \top_k y \iff \mathrm{val}_g(y) \leftarrow \gamma_k\big(\mathrm{val}_g(x)\big) \tag{F.1}$$

L'ensemble des arcs du graphe pyramidal $\overline{A}_{R(V)} = A_{R(V)} \cup A'_{R(V)} \cup A''_{R(V)}$ est donc aussi l'ensemble des couples de sommets de $S_{\Sigma,V}$ dont les sous-graphes desquels ils sont racine vérifient l'une des q relations $\top_k$. En fait, on a la partition $\overline{A}_{R(V)} = \overline{A}^1_{R(V)} \oplus \ldots \overline{A}^q_{R(V)}$ où chaque $\overline{A}^k_{R(V)}$ est l'ensemble des couples de sous-graphes vérifiant la relation $\top_k$. Par conséquent, tout arc $s_i \xrightarrow{f} s_j$ appartient à l'un des $\overline{A}^k_{R(V)}$ et à lui seul.

Montrons maintenant que l'injection K est un homéomorphisme de l'espace $(S_{\Sigma,V}, \bigcup_k \top_k)$ dans l'espace $(K(S_{\Sigma,V}), \bigcup_k \bot_k)$ en raison du diagramme de la figure F.1 qui définit K opératoirement.

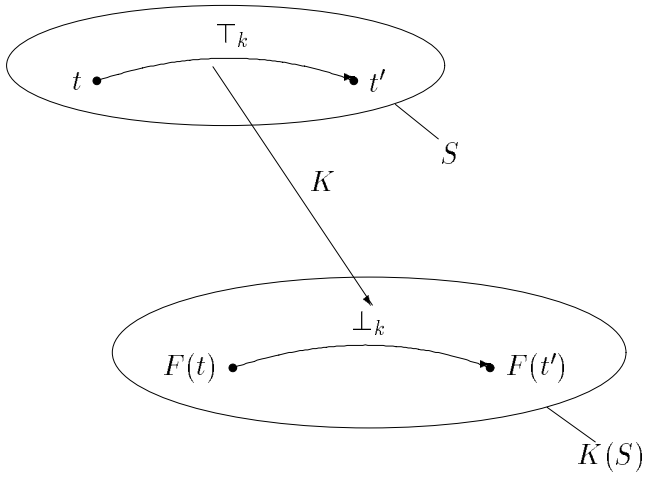

FIG. F.1: Définition graphique du foncteur K

Les relations $\bot_k$ expriment les mêmes fonctions que les relations $\top_k$, mais cette fois-ci entre deux pyramides. Ainsi, pour deux pyramides p et p' de $K(S_{\Sigma,V})$, on a :

$$p \bot_k p' \iff \mathrm{val}_p(p') \leftarrow \gamma_k\big(\mathrm{val}_p(p)\big) \tag{F.2}$$

Cela signifie que les relations $\top_k$ et $\bot_k$ énoncent les mêmes règles de calcul concernant les valeurs d'activité, mais qu'elles opèrent entre des éléments de natures différentes : entre des sommets de graphes pyramidaux pour les $\top_k$ et entre des pyramides pour les $\bot_k$.

A présent, dire que K est un homéomorphisme de $(S_{\Sigma,V}, \bigcup_k \top_k)$ dans $(K(S_{\Sigma,V}), \bigcup_k \bot_k)$ signifie que pour tous sommets s et s' de $S_{\Sigma,V}$, on a :

$$K(t \top_k t') = K(t) \bot_k K(t')$$

On va montrer cette égalité pour une injection partielle $K_{|k}$ qui opère uniquement sur le sous-graphe de $G_{\Sigma,V,R(V)}$ dont l'ensemble des arcs est réduit à $\overline{A}^k_{R(V)}$. Pour cela, on part de l'égalité déduite de F.1 :

$$t \top_k t' \iff \mathrm{val}_g(t') \leftarrow \gamma_k(\mathrm{val}_g(t))$$

Afin de mieux détailler cette égalité, nous allons représenter la relation entre t et t' non plus comme faisant partie de $(S^2_{\Sigma,V}, \top_k)$, mais comme élément d'un autre ensemble équivalent :

$$\begin{array}{ccc} t \top_k t' & = & \big(t, t', \gamma_k, \mathrm{val}_g\big) \\ \in & & \in \\ (S^2_{\Sigma,V}, \top_k) & & (S^2_{\Sigma,V}, \gamma_k, \mathrm{val}_g) \end{array}$$

L'égalité entre $\top_k$ et $(\gamma_k, \mathrm{val}_g)$ est évidente du fait de la formule F.1.

On peut maintenant appliquer directement le foncteur K :

$$\begin{array}{rcl} K_{|k}(t \top_k t') & = & K_{|k}(t, t', \gamma_k, \mathrm{val}_g) \\ & = & \big(K_{|k}(t), K_{|k}(t'), K_{|k}(\gamma_k), K|k(\mathrm{val}_g)\big) \\ & = & (F(G^t_{\Sigma,V,R(V)}), F(G^{t'}_{\Sigma,V,R(V)}), \gamma_k, \mathrm{val}_p) \text{ car } K_{|k} \text{ est un foncteur} \\ & = & F(t) \bot_k F(t') \end{array}$$

La dernière égalité est due à la même identité évidente que précédemment, mais cette fois-ci entre $(K(S_{\Sigma,V})^2, \bot_k)$ et $(K(S_{\Sigma,V})^2, \gamma_k, \mathrm{val}_p)$ en raison de la formule F.2.

Le foncteur partiel $K_{|k}$ est donc bien un homéomorphisme. Comme ceci est vrai pour tout k et que la partition $\overline{A}_{R(V)} = \overline{A}^1_{R(V)} \oplus \ldots \overline{A}^q_{R(V)}$ induit une partition semblable sur les foncteurs partiels $K_{|1}$ à $K_{|q}$ (sous-entendu il ne peut y avoir d'élément de $\overline{A}_{R(V)}$ qui ait une image à la fois par $K_{|i}$ et par $K_{|j}$), on a donc bien : K est un homéomorphisme de $(S_{\Sigma,V}, \bigcup_k \top_k)$ dans $(K(S_{\Sigma,V}), \bigcup_k \bot_k)$.

Inversement, on peut définir un foncteur K' tel que pour pyramide $P_{\Sigma,V,R(V)} = K(G_{\Sigma,V,R(V)})$ on a $K'(P_{\Sigma,V,R(V)}) = G_{\Sigma,V,R(V)}$.

Si on note $P_{\Sigma,V,R(V)} = \big\langle S_{\Sigma,V,R(V)}, A_{R(V)}, E_{\Sigma,V} \big\rangle$, alors on a :

$$K'(P_{\Sigma,V,R(V)}) = \big\langle K'_0(P_{\Sigma,V,R(V)}), K'_1(P_{\Sigma,V,R(V)}), K'_2(P_{\Sigma,V,R(V)}), K'_3(P_{\Sigma,V,R(V)}) \big\rangle$$

avec :

— l'ensemble des sommets du graphe est $K'_0(P_{\Sigma,V,R(V)}) = \big\{x, F^{-1}(s_1), \ldots, F^{-1}(s_p)\big\}$ où les s_i sont les sommets de la pyramide, et x est un nouveau sommet (du graphe pyramidal) tel que $\mathrm{etiq}(x) = \mathrm{etiq}(P_{\Sigma,V,R(V)})$ et $\mathrm{val}(x) = \mathrm{val}(P_{\Sigma,V,R(V)})$;

— l'ensemble $K'_1(P_{\Sigma,V,R(V)})$ des arcs de l'arbre n-aire sous-jacent au graphe pyramidal est déduit des relations d'appartenance au sein de la pyramide : $\Big(a \xrightarrow{f} b\Big) \in K'_1(P_{\Sigma,V,R(V)}) \Rightarrow a = x$ et $y \in S_{\Sigma,V,R(V)}$ ou bien $\Big(a \xrightarrow{f} b\Big) \in F_1^{-1}(y)$ où y est un sommet de la pyramide et F_1^{-1} est déduite de F^{-1} de la même façon que K'_1 est déduite de K' ;

— l'ensemble des arcs verticaux "montantst't' du graphe pyramidal $K'_2(P_{\Sigma,V,R(V)})$ est déduit de $K'_1(P_{\Sigma,V,R(V)})$: si $\Big(a \xrightarrow{f} b\Big) \in K'_1(P_{\Sigma,V,R(V)})$ alors $\exists g, \Big(b \xrightarrow{g} a\Big) \in K'_2(P_{\Sigma,V,R(V)})$;

— l'ensemble des arcs horizontaux du graphe pyramidal est l'union récursive des relations existant entre les pyramides incluses dans $P_{\Sigma,V,R(V)}$: $\left(a \xrightarrow{f} b\right) \in K'_3(P_{\Sigma,V,R(V)}) \Rightarrow \left(s_i \xrightarrow{f} s_j\right) \in A_{R(V)}$ et $a = F^{-1}(s_i), b = F^{-1}(s_j)$ ou bien $\left(a \xrightarrow{f} b\right)$ est un des arcs horizontaux de l'un des $F^{-1}(s_i)$.

Le foncteur K' est un homéomorphisme pour les mêmes raisons que K est un homéomorphisme, et de plus $K' = K^{-1}$. K est donc une bijection entre l'ensemble des graphes pyramidaux de profondeur $n+1$ et un sous-ensemble de l'ensemble des pyramides que nous noterons $\mathcal{P}^{n+1}_{\Sigma,V,R(V)}$.

On peut donc définir une bijection F' de l'ensemble des graphes pyramidaux de profondeur au plus $n+1$ dans l'ensemble des pyramides par adjonction de F et de K (en passant par les duaux où $F'^* = F^* \cup K^*$).

Pour simplifier les notations nous confondrons F' avec F (puisque F' est une extension de F). Ainsi, en partant de l'hypothèse de la bijectivité de F entre graphes pyramidaux et pyramides de profondeur au plus n, nous sommes arrivés à la conclusion que F est également bijective pour une profondeur au plus $n+1$.

F.2.3 Conclusion

Finalement, par récurrence sur n, on a la preuve qu'il existe bien une bijection entre les graphes pyramidaux et les pyramides. □