Architecture modulaire portable pour la génération du langage naturel en dialogue homme-machine

Vladimir POPESCU[1,2]
[1] Laboratoire d'Informatique de Grenoble, France
[2] Université « Politehnica » de Bucarest, Roumanie
Vladimir.Popescu@imag.fr

Résumé. La génération du langage naturel pour le dialogue oral homme-machine pose des contraintes spécifiques, telles que la spontanéité et le caractère fragmenté des énoncés, les types des locuteurs ou les contraintes de temps de réponse de la part du système. Dans ce contexte, le problème d'une architecture rigoureusement spécifiée se pose, autant au niveau des étapes de traitement et des modules impliqués, qu'au niveau des interfaces entre ces modules. Afin de permettre une liberté quasi-totale à l'égard des démarches théoriques, une telle architecture doit être à la fois modulaire (c'est-à-dire, permettre l'indépendance des niveaux de traitement les uns des autres) et portable (c'est-à-dire, permettre l'interopérabilité avec des modules conçus selon des architectures standard en génération du langage naturel, telles que le modèle RAGS - « Reference Architecture for Generation Systems »). Ainsi, dans cet article on présente de manière concise l'architecture proposée, la comparant ensuite au modèle RAGS, pour argumenter les choix opérés en conception. Dans un second temps, la portabilité de l'architecture sera décrite à travers un exemple étendu, dont la généralité réside dans l'obtention d'un ensemble de règles permettant de plonger automatiquement les représentations des informations de notre architecture vers le format du modèle RAGS et inversement. Finalement, un ensemble de conclusions et perspectives clôturera l'article.

Abstract. Natural language generation for human-computer dialogue imposes specific constraints, such as the spontaneous and fragmented character of the utterances, speaker types or constraints related to the system's time of response. In this context, the issue of a thouroughly specified architecture stems naturally, with respect to the processing stages in the modules involved and to the interfaces between these modules as well. In order to allow for a quasi-total freedom concerning the theoretical principles driving the processing stages, such an architecture must be modular (i.e., allowing the independence of the modules of each other) and portable (i.e., allowing a certain interoperability between modules designed following this architecture and existing modules, designed following standard, reference architectures, such as the RAGS model). Thus, in this paper firstly the proposed architecture will be presented in a concise manner, comparing it then to the RAGS model and arguing for the design choices being made. Afterwards, the portability of the architecture will be described, via an extended example whose general character resides in the fact that a set of rules are obtained, that allow automatic translations between representation formats in our architecture and in the RAGS model, in both ways. Finally, a set of conclusions and pointers to further work end up the paper.

Mots-clés : génération, dialogue, architecture modulaire, portabilité, XML.

Keywords: generation, dialogue, modular architecture, portability, XML.

1 Introduction

Nos recherches se situent dans le cadre de la génération du langage naturel pour le dialogue oral homme-machine et concernent le développement d'un module de génération du langage naturel pour donner un caractère aussi « naturel » et expressif que possible aux réponses langagières du système face aux requêtes des usagers. Ce travail poursuit ainsi des recherches commencées depuis plusieurs années (Imberdis & Caelen, 1997). Le problème n'est pas simple, car la plupart des générateurs textuels existants sont conçus pour des situations de monologue, et il s'y ajoutent le caractère spontané et fragmenté du dialogue, auquel des contraintes de pertinence, expressivité et temps de réponse se conjuguent (McTear, 2002).

Ainsi, nous considérons (Popescu *et al.*, 2007) que la réponse du système se situe à cinq niveaux auxquels le langage naturel peut être « produit » par un système de dialogue : (i) le niveau **logique** instantié dans un contrôleur de dialogue et ne faisant pas partie du générateur, mais fournissant l'intention communicationnelle à mettre sous forme linguistique, (ii) le niveau **pragmatique** gérant les aspects liés à l'expressivité des énoncés et à leur pertinence par rapport au contexte dialogique, (iii) le niveau **linguistique** produisant le texte pour l'intention communicationnelle, (iv) le niveau **expressif** calculant la forme finale de l'énoncé et la prosodie, et (v) le niveau **acoustique** réalisant la synthèse de la parole proprement dite.

L'architecture conçue pour la génération dans le cadre du dialogue homme-machine part d'un ensemble de principes :

1. prendre en compte les aspects pragmatiques (rhétoriques - structuration discursive et expressifs - valences émotionnelles) et la gestion des tours de parole (dialogue oral spontané), ainsi que les particularités des locuteurs ;
2. considérer surtout les performances des agents du dialogue (en dépit des normes de compétence qui varient d'un contexte (social, situationnel, etc.) à l'autre et ne sont donc pas génériques) ;
3. s'appuyer sur un corpus de dialogues réels entre humains et entre homme et machine (en dépit des prescriptions grammaticales fixées a priori) pour contrôler les aspects linguistiques en génération ;
4. rendre les traitements appropriés à un fonctionnement en temps réel (et donc éviter des processus d'inférence relativement coûteux) ;
5. diminuer autant que possible la dépendance à la tâche et à la langue, en permettant des paramétrages aisés ;
6. rendre les niveaux de traitement de la parole aussi indépendants que possible des traitements purement textuels.

Le premier point des désidérata ci-dessus est réalisé dans le générateur pragmatique, où les aspects rhétoriques-discursifs sont gérés par l'utilisation adaptée de la théorie SDRT (« Segmented Discourse Representation Theory ») (Asher & Lascarides, 2003), tandis que les aspects expressifs au niveau de chaque énoncé sont gérés par le contrôle du « degré de puissance de la force illocutoire ». De plus, on prévoit la prise en compte, dans le générateur pragmatique, d'un doublet d'attributs caractérisant chaque partenaire du dialogue (en termes de niveau de familiarité par rapport au dialogue courant et de relation sociale de l'un avec l'autre).

Le cinquième point est réalisé par le fait que, autant au niveau pragmatique que linguistique, les méthodes utilisées sont indépendantes de la tâche et de la langue (supposant en fin de compte

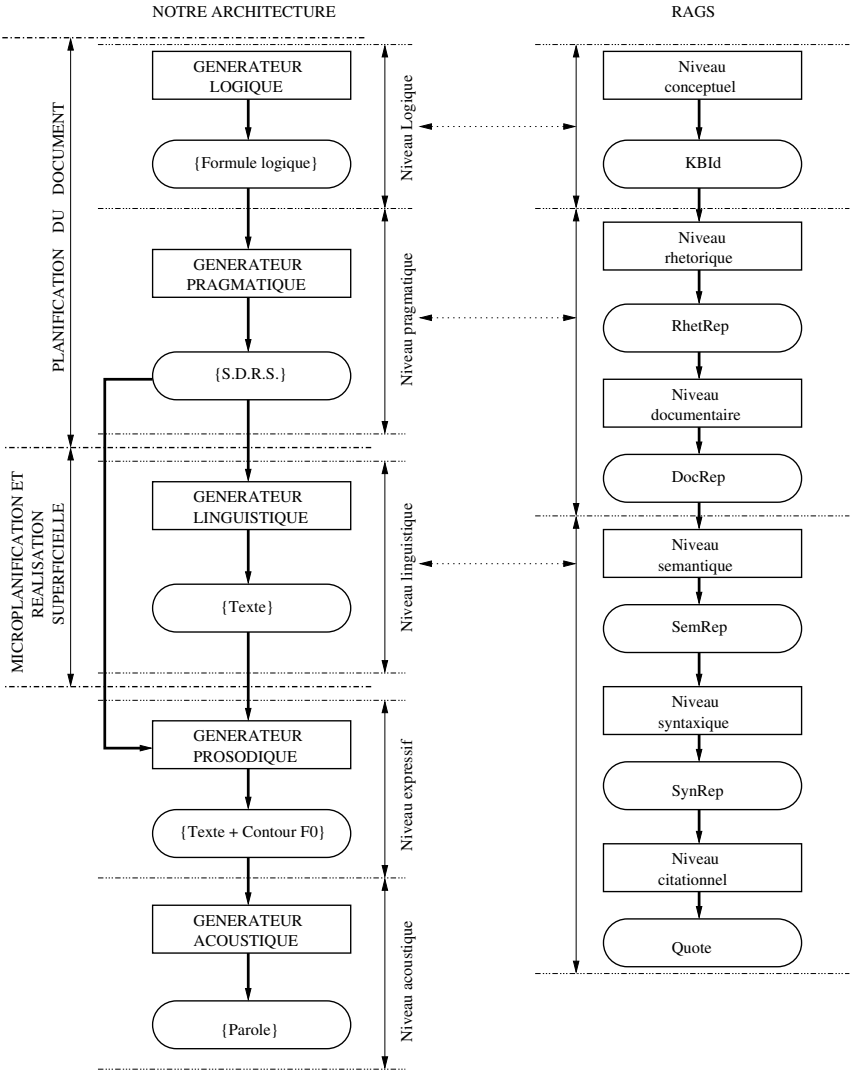

FIG. 1 – Architecture modulaire de génération et correspondances avec le modèle RAGS

des traitements du type appariement des graphes) et paramétrables dans une langue et pour une tâche données.

Le sixième point est réalisé par l'utilisation d'un format standard pour la représentation des connaissances (XML) et d'une théorie de représentation pragmatique-discursive dont le formalisme « interne » n'est pas pris en compte (la SDRT - « Segmented Discourse Representation

Theory », dont on ne prend en compte que les relations rhétoriques et leurs sémantiques *informelles*).

L'idée de concevoir une architecture pour la génération du langage naturel a été déjà énoncée dans plusieurs études, comme celle de Reiter et Dale (Reiter & Dale, 2000) et l'architecture RAGS (Mellish *et al.*, 2006). Mais aucun de ces deux modèles ne traite le dialogue homme-machine. Il existe peu de travaux pour définir une architecture adaptée pour la génération en dialogue (Imberdis & Caelen, 1997), (McTear, 2002).

D'autres travaux ont abordé la génération en dialogue de diverses façons, dont les plus notables sont, à notre sens, ceux d'Amanda Stent (Stent, 2001), de Mariet Theune (Theune, 2000) et de Matthew Stone (Stone, 1998), puisqu'ils sont explicitement concernés par le dialogue oral homme-machine, aboutissant en même temps à des systèmes fonctionnels. Ces démarches ne sont pas génériques, sont sans référence à un « standard » en génération (tels que l'architecture de Reiter et Dale ou le modèle RAGS). Cela implique le manque de portabilité et de réutilisation de ces systèmes. Notre architecture proposée ici est fortement compatible à la fois avec le propos de Reiter et Dale et avec le modèle RAGS.

Pour l'utilisation de la SDRT en génération en situations de monologues, les travaux de Laurence Danlos et son équipe peuvent être citées (Danlos *et al.*, 2001) ; pour des extensions de la SDRT pour l'interprétation des dialogues, les travaux de Laurent Prévot (Maudet *et al.*, 2004) et son équipe sont intéressants, mais ne concernent pas la génération. Dans ce contexte, les travaux assumés par notre projet essaient de renforcer les recherches en génération pour le dialogue homme-machine finalisé.

2 Architecture modulaire compatible avec les « standards » en génération

Le schéma global de l'architecture est illustré dans la figure 1, où on met en évidence la relation entre les niveaux de traitement, les modules (générateurs) et les connaissances échangées entre les *modules* et implicitement entre les niveaux.

Sur cette figure les modules de traitement sont représentés par des rectangles et les structures de données par des ellipses. A droite, les flèches doubles montrent les niveaux de traitement, tandis qu'à gauche, les flèches situent l'architecture proposée ici par rapport à l'architecture de référence de Reiter et Dale (Reiter & Dale, 2000), qui représente une première spécification, plutôt théorique, d'une architecture pour la génération du langage naturel.

Les interfaces entre les modules de génération sont spécifiées en XML et les choix de conception pour notre architecture ont été décrites de manière étendue dans (Popescu *et al.*, 2007). Le modèle RAGS est décrit dans (Mellish *et al.*, 2006). Les entités dénommées « primitives abstraites » à un niveau de traitement dans la description du modèle RAGS ci-dessus sont en fait des pointeurs renvoyant vers les représentations les plus complexes au niveau inférieur ; par exemple, la primitive RhetLeaf au niveau rhétorique renvoie vers une structure sémantique, SemRep.

Le modèle RAGS spécifie une architecture « pipeline », permettant des raccourcis entre les niveaux de représentation, via les flèches *non-locales* (Mellish *et al.*, 2006). Donc, la double incidence du niveau expressif à d'autres niveaux de traitement est compatible en théorie et en

pratique avec RAGS.

Les correspondances entre RAGS et notre architecture sont montrées dans la figure 1, où les connaissances transférées dans le modèle RAGS sont des primitives (dans le cas de KBId et Quote) où des représentations dérivées (pour le reste - cf. la discussion ci-dessous).

Notre architecture

- **Niveau Logique** - Ce niveau de traitement ne fait pas partie à vrai dire de la composante de génération d'un système de dialogue, mais on le précise ici en raison de compatibilité avec les architectures existantes dans le domaine de la génération ; il fournit une forme logique correspondante à l'intention communicationnelle à mettre sous forme linguistique, prenant en entrée une formule logique correspondante à l'intention communicationnelle d'un énoncé provenu de l'usager ;
- **Niveau pragmatique** - Ce niveau de traitement est le premier à faire vraiment partie du système de génération ; il utilise la SDRT et les actes de langage pour fournir une représentation discursive du dialogue courant, prenant en entrée la formule logique pour l'énoncé à engendrer ;
- **Niveau linguistique** - Ce niveau de traitement opère la mise en forme linguistique du message à générer par le système et prend en entrée la structure discursive pour le dialogue courant, fournissant le texte correspondant à l'énoncé matérialisant la réponse du système ;
- **Niveau expressif** - Ce niveau de traitement ajoute un degré d'expressivité au texte engendré au niveau linguistique, prenant en entrée la structure discursive pour fournir en sortie les paramètres prosodiques appropriés associés au texte à générer ; l'expressivité est considérée seulement à l'égard de la qualité du signal vocal synthétisable à partir du texte ;
- **Niveau acoustique** - Ce niveau réalise la synthèse vocale proprement dite, en prenant à l'entrée un texte annoté pour une gestion fine de la prosodie pour fournir en sortie la parole synthétisée ;

Le modèle RAGS

- **Niveau conceptuel** - Ce niveau de traitement fournit des représentations non-linguistiques des informations à communiquer (par exemple des formules logiques) ; ces représentations conceptuelles sont manipulées via des pointeurs vers des entités primitives dans une base de connaissances et appelées KBId (« Knowledge Base Identifier ») ;
- **Niveau rhétorique** - Ce niveau de traitement définit des relations discursives entre les énoncés enchaînés dans un discours ; la manière dont les relations sont réalisées est spécifiée via les primitives abstraites RhetRel (pour la relation) et RhetLeaf (pour les arguments de la relation) à partir desquelles les structures composites RhetRepSeq et RhetRep sont construites ;
- **Niveau documentaire** - Ce niveau de traitement concerne la disposition physique des énoncés (voire typage du texte) à produire et les structures de données fournies sont DocAttr, DocLeaf, DocRepSeq et DocRep, construites à partir des primitives abstraites DocFeat, DocAtom et DocLeaf ;
- **Niveau sémantique** - Ce niveau de traitement peut être combiné avec le niveau rhétorique, ainsi qu'au niveau sémantique on code à la fois les propositions atomiques (cf. celles trouvées dans les structures rhétoriques) et les organisations structurées de propositions, pour l'entrée aux générateurs dits « à entrée sémantique » ; les structures de données fournies par ce niveau, SemRep, ScopedSemRep, SemType, SemAttr,Scoping et ScopeConstr sont construites à partir des primitives abstraites DR, SemConstant, SemPred et ScopeRel ;
- **Niveau syntaxique** - Ce niveau fournit une représentation syntaxique abstraite qui ne spécifie pas l'ordre des mots à l'intérieur des énoncés ; par rapport à la représentation sémantique, la structure syntaxique comprend les fonctions grammaticales des mots et est représentée par les types SynRep, FVM (de « Feature-value matrix », matrice attribut-valeur), SynArg, et Adj, construites à partir des primitives nil, SynFun, SynFeat et SynAtom ;
- **Niveau citationnel** - Ce niveau concerne surtout les fragments fixés de texte (les structures figées) incluses, sans modification, dans la sortie du système de génération ; pour la représentation des données à ce niveau seule une primitive abstraite existe, Quote ;

L'architecture proposée ici et son parallèle, l'architecture RAGS sont comparés ci-dessous.

Ainsi, le niveau logique de notre architecture correspond au niveau conceptuel du modèle RAGS seulement du point de vue de la génération, car le générateur logique (en fait, le contrôleur de dialogue) a un rôle plus étendu que le niveau conceptuel de RAGS. C'est pour cela que le niveau logique ne fait pas à vrai dire partie de notre architecture, mais représente seulement l'interface du système de génération au contexte dialogique.

Le niveau pragmatique de notre architecture correspond au niveau rhétorique de RAGS et au niveau documentaire de ce dernier, car notre générateur pragmatique assure la situation et la cohérence rhétorique de l'acte de langage à engendrer, en fournissant en même temps une représentation qui constitue une structuration discursive.

Le niveau linguistique de notre architecture correspond à trois niveaux de RAGS : les niveaux

sémantique, syntaxique et citationnel.

Quant aux niveaux expressif et acoustique de notre architecture, ceux-ci n'ont pas de correspondants en RAGS car le côté prosodie et synthèse de parole représente un aspect de la génération dont RAGS ne rend pas compte. RAGS est concerné seulement avec l'obtention du texte comme résultat final du processus de génération.

En conclusion, on observe que l'architecture, bien qu'elle s'appuie sur d'autres architectures de référence, soit plus théoriques comme celle de Reiter et Dale, soit plus orientées logiciel, comme RAGS, reste particulière dans les détails fins.

3 Architecture portable

Nous discutons maintenant de l'isomorphisme entre les représentations de RAGS et les représentations de notre architecture, en nous appuyant sur un exemple de représentations dans le modèle RAGS et dans notre architecture.

Nous illustrons cela autour d'un exemple pour la représentation rhétorique correspondante à un dialogue court entre deux locuteurs, ayant comme objet l'établissement d'un rendez-vous :

$U : \pi_1$: Quand es-tu disponible aujourd'hui ?

$M : \pi_2$: Je suis disponible aujourd'hui dès 16h !

Ainsi, en employant une formalisation rhétorique à la SDRT (Asher & Lascarides, 2003), on suppose que les générateurs doivent représenter la structuration rhétorique du dialogue indiqué ci-dessus.

Pour ce texte, la représentation rhétorique RhetRep que le niveau rhétorique de RAGS construit dans son langage interne est montrée dans la figure 2, à côté de la représentation issue de notre architecture, dans le cadre de la SDRT. Les conventions de représentation pour le RAGS sont les suivantes (Mellish *et al.*, 2006) : (i) les noeuds sont des *objets* qui représentent des structures et des sous-structures (types primitifs) ; à chaque objet on associe un type RAGS (de ceux définis dans le modèle) ; de ces types, seulement les types primitifs spécifient des informations se reportant à la théorie choisie pour *concrétiser* l'architecture abstraite de RAGS, et (ii) les *flèches* entre les objets représentent les relations entre ces derniers ; pour chaque flèche il existe une et seulement une source et une cible ; en même temps, chaque flèche est désignée par une étiquette unique. De plus, les flèches sont typées, dans le sens que pour chaque étiquette il y a un unique *type* pour la source et un type toujours unique pour la cible ; ces types sont précisés lors de la définition de chaque flèche, donc, immuables. En revanche, pour la représentation de notre architecture, la notation « > » sépare deux éléments en structure arborescente, tandis que « : : » sépare un élément (à gauche) d'un attribut des siens propres.

On peut voir le texte ci-dessus relié par la relation discursive SDRT QAP (« Question - Answer Pair »), dont la représentation RAGS est illustrée dans la figure 2.

La représentation XML correspondant à la représentation de la figure 2, pour cet exemple, est donnée dans la figure 3.

La figure 4 montre un ensemble de règles permettant le passage de la représentation rhétorique de RAGS à la représentation utilisée en sortie du module pragmatique de notre architecture ; on utilise les mêmes notations concernant « > » et « : : ».

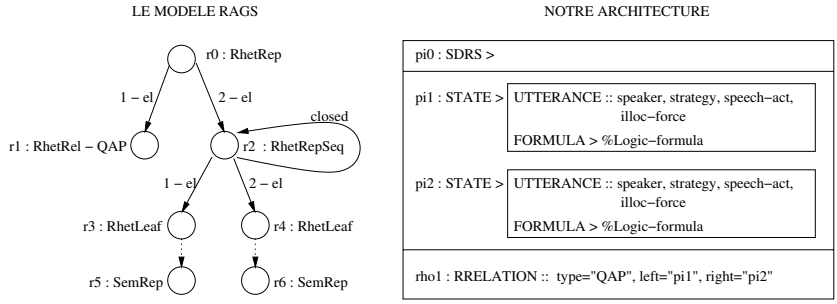

FIG. 2 – Exemple de représentation rhétorique dans RAGS et dans notre architecture pour une instance de dialogue simple

Cet ensemble de règles montre que pour 23 des 34 règles il existe une correspondance entre les éléments et attributs dans la représentation RAGS et dans notre architecture, donc il y a une superposition de 2/3 entre les représentations dans les deux architectures au sens où la représentation dans notre architecture couvre 2/3 de la représentation équivalente en RAGS. Le reste comprend surtout des informations d'identification interne des structures de données, ce qui n'est pas nécessaire dans notre architecture, car elle est conçue pour le dialogue homme-machine où le système doit générer un tour de parole à un instant donné et non pas élaborer des discours monologiques étendus, où un plus fort repérage des structures linguistiques partielles est nécessaire dans le temps.

Procédant à l'envers et construisant un ensemble de règles permettant de « traduire » la spécification XML des informations rhétoriques échangées entre les modules de génération pragmatique et linguistique en représentation selon le modèle RAGS on obtient une couverture de 5/6 selon un ensemble de règles semblables à ceux juste présentées ci-dessus. Le reste de 1/6 relève du manque de « transport » vers la représentation RAGS des marqueurs spécifiques au dialogue (l'identité des locuteurs et les marqueurs pragmatiques dialogiques - stratégie, acte de langage et force illocutoire). Ces informations peuvent cependant être récupérées au niveau des identifiants en RAGS, qu'une structure parallèle réalisant les correspondances appropriées peut accompagner, en tant que connaissances *statiques* sous forme de tableau de correspondances.

4 Conclusion

Dans cet article nous avons présenté une vue d'ensemble d'architecture **générique** (dans le sens que les modules de traitement sont indépendants des *interfaces* entre eux), **modulaire** (dans le sens que les traitements dans un module sont indépendants des traitements dans les autres modules) et **portable** (dans le sens que même au niveau des interfaces les modules sont indépendants pourvu qu'on plonge les représentations des connaissances à chaque niveau de traitement dans une architecture standard en génération du langage naturel, telle que le modèle RAGS).

Plus précisément, un exemple étendu où la représentation de l'interface entre les modules pragmatique et linguistique dans notre architecture, comportant des informations rhétoriques échan-

LE MODELE RAGS

```
<RAGS>
 <RhetRep ident="id1" type="tuple"
          length="2">
  <RhetRel name="QAP"/>
  <RhetRepSeq ident="i2" type="sequence">
   <RhetLeaf ident="ID3"/>
   <RhetLeaf ident="ID4"/>
  </RhetRepSeq>
 </RhetRep>
 <arrow name="refers_to" source="ID3"
        target="ID5"/>
 <arrow name="refers_to" source="ID4"
        target="ID6"/>
 <SemRep ident="ID5" type="tuple"
         length="3">
  <DR name="r5"/>
  <SemType type="set">
   <SemPred name="available"/>
  </SemType>
  <SemAttr type="functional">
   <SemRoleRep>
    <SemRole name="actor"/>
    <SemConstant ident="C1"
                 name="patient"/>
   </SemRoleRep>
   <SemRoleRep>
    <SemRole name="actee">
     <SemRep ident="s2" type="tuple"
             length="3">
      <DR name="r7"/>
      <SemType type="set">
       <SemPred name="tu"/>
      </SemType>
      <SemAttr type="functional">
       <SemRoleRep>
        <SemRole name="person"/>
        <xref idref="C1"/>
       </SemRoleRep>
      </SemAttr>
     </SemRole>
    </SemRep>
   </SemRoleRep>
  </SemAttr>
 </SemRep>
 <SemRep ident="ID6" type="tuple"
         length="3">
  <DR name="r6"/>
  <SemType type="set">
   <SemPred name="16:00"/>
  </SemType>
  <SemAttr type="functional">
   <SemRoleRep>
    <SemRole name="subject"/>
    <xref idref="S2"/>
   </SemRoleRep>
  </SemAttr>
 </SemRep>
</RAGS>
```

NOTRE ARCHITECTURE

```
<SDRS label="pi0" nstates="2" nrels="1">
 <STATE label="pi1" type="UTT">
  <UTTERANCE speaker="A" strategy="K2"
             speech-act="FFS" illoc-force="36"/>
  <FORMULA>
   <quant name="exists" variable="X">
    <type>person</type>
    <quant name="exists" variable="Y">
     <type>time</type>
     <conn name="and">
      <pred name="equals">
       <term name="X"/> <term name="A"/>
      </pred>
      <pred name="equals">
       <term name="Y"/> <term name="?"/>
      </pred>
      <pred name="available">
       <term name="X"> <term name="Y">
      </pred>
     </conn>
    </quant>
   </quant>
  </FORMULA>
 </STATE>
 <STATE label="pi2" type="UTT">
  <UTTERANCE speaker="B" strategy="K2"
             speech-act="FS" illoc-force="54"/>
  <FORMULA>
   <quant name="exists" variable="X">
    <type>person</type>
    <quant name="exists" variable="Y">
     <type>time</type>
     <quant name="exists" variable="Z">
      <type>offset</type>
      <quant name="exists" variable="T">
       <type>direction</type>
       <conn name="and">
        <pred name="equals">
         <term name="X"/> <term name="A"/>
        </pred>
        <pred name="equals">
         <term name="Y"/> <term name="16:00+Z"/>
        </pred>
        <pred name="equals">
         <term name="Z"/> <term name="0"/>
        </pred>
        <pred name="equals">
         <term name="T"/> <term name="t+"/>
        </pred>
       </conn>
      </quant>
     </quant>
    </quant>
   </quant>
  </FORMULA>
 </STATE>
 <RRELATION label="rho1" type="QAP"  left="pi1" right="pi2"/>
</SDRS>
```

FIG. 3 – Représentations XML des informations rhétoriques, dans le modèle RAGS et dans notre architecture

1.	RAGS	→	∅
2.	RAGS > RhetRep	→	SDRS
3.	RAGS > RhetRep : : ident	→	SDRS : : label
4.	RAGS > RhetRep : : type	→	∅
5.	RAGS > RhetRep : : length	→	SDRS : : nstates
6.	RAGS > RhetRep > RhetRel	→	SDRS > RRELATION
7.	RAGS > RhetRep > RhetRel : : name	→	SDRS > RRELATION : : type
8.	RAGS > RhetRep > RhetRepSeq	→	SDRS > RRELATION
9.	RAGS > RhetRep > RhetRepSeq : : ident	→	SDRS > RRELATION : : label
10.	RAGS > RhetRep > RhetRepSeq : : type	→	∅
11.	RAGS > RhetRep > RhetRepSeq > RhetLeaf	→	SDRS > STATE
12.	RAGS > RhetRep > RhetRepSeq > RhetLeaf : : ident	→	SDRS > STATE : : label | SDRS > RRELATION : : left | SDRS > RRELATION : : right
13.	RAGS > arrow	→	∅
14.	RAGS > arrow : : name	→	∅
15.	RAGS > arrow : : source	→	∅
16.	RAGS > arrow : : destination	→	∅
17.	RAGS > SemRep	→	SDRS > STATE | SDRS > STATE > FORMULA
18.	RAGS > SemRep : : ident	→	SDRS > STATE : : label
19.	RAGS > SemRep : : type	→	SDRS > STATE : : type
20.	RAGS > SemRep : : length	→	∅
21.	RAGS > SemRep > DR	→	SDRS > STATE
22.	RAGS > SemRep > DR : : name	→	SDRS > STATE : : label
23.	RAGS > SemRep > SemType	→	SDRS > STATE | SDRS > STATE > FORMULA
24.	RAGS > SemRep > SemType : : type	→	SDRS > STATE : : type
25.	RAGS > SemRep > SemType > SemPred	→	SDRS > STATE > FORMULA > quant > conn > pred
26.	RAGS > SemRep > SemType > SemPred : : name	→	SDRS > STATE > FORMULA > quant > conn > pred : : name
27.	RAGS > SemRep > SemAttr	→	∅
28.	RAGS > SemRep > SemAttr : : type	→	∅
29.	RAGS > SemRep > SemAttr > SemRoleRep	→	SDRS > STATE > FORMULA | SDRS > STATE > FORMULA > quant | SDRS > STATE > FORMULA > quant > conn
30.	RAGS > SemRep > SemAttr > SemRoleRep > SemRole	→	SDRS > STATE > FORMULA > quant > conn > pred > term | SDRS > STATE > FORMULA > quant
31.	RAGS > SemRep > SemAttr > SemRoleRep > SemRole : : name	→	SDRS > STATE > FORMULA > quant > type
32.	RAGS > SemRep > SemAttr > SemRoleRep > Constant	→	∅
33.	RAGS > SemRep > SemAttr > SemRoleRep > Constant : : ident	→	SDRS > STATE > FORMULA > quant > variable
34.	RAGS > SemRep > SemAttr > SemRoleRep > Constant : : name	→	SDRS > STATE > FORMULA > quant > type

FIG. 4 – Règles de transformation du format XML RAGS vers celui de notre architecture

gées, a été plongée dans la représentation rhétorique, située au niveau conceptuel de traitement, dans le modèle RAGS. Lors de cet exercice, rendu possible via un ensemble de règles transformationnelles, on a constaté que la « traduction » dans les deux formats de représentation n'est pas totale, mais conserve les éléments pertinents par rapport à la nature de l'échange, dans les deux sens, de notre architecture vers RAGS et à l'inverse. Ce résultat nous amène a démontrer la portabilité annoncée : les modules dans notre architecture peuvent être combinés ou « entrelacés » avec des modules conçus suivant les spécifications du modèle RAGS ; par exemple, on pourrait construire un système de génération du langage naturel ayant les niveaux logique et pragmatique de notre architecture, ayant aussi un niveau linguistique éclaté en niveaux sémantique, syntaxique et citationnel d'un module de génération suivant les spécifications du modèle RAGS.

On a donc présenté une architecture qui étend un modèle de génération automatique des textes (RAGS) au dialogue oral homme-machine, en gardant, dans une mesure importante, la compatibilité entre les deux.

Nous envisageons de concevoir des algorithmes pour l'obtention automatique des règles de traduction entre notre architecture et le modèle RAGS ; ceci peut se réaliser en principe par l'appariement d'arbres XML pour les représentations dans les deux architectures ; la suite des

transformations d'un arbre vers l'autre constitue l'ensemble des règles. Evidemment, les démarches décrites dans cet article concernant l'interface entre les niveaux pragmatique et linguistique dans notre architecture et, respectivement, l'interface entre les niveaux rhétorique et sémantique dans le modèle RAGS doivent être étendues aux autres interfaces entre les modules des deux architectures.

Remerciement

L'auteur remercie vivement Jean Caelen, du Laboratoire d'Informatique de Grenoble, pour ses conseils attentifs et pour son apport aux travaux présentés dans cet article.

Références

ASHER N. & LASCARIDES A. (2003). *Logics of Conversation*. Cambridge University Press.

DANLOS L., GAIFFE B. & ROUSSARIE L. (2001). Document structuring à la sdrt. In ACL, Ed., *Proceedings of the 8th European Workshop on Natural Language Generation EWNLG 2001*.

IMBERDIS L. & CAELEN J. (1997). Génération d'actes illocutoires pour le dialogue. In *Actes du Colloque Génération automatique du texte GAT'97*, Grenoble.

MAUDET N., MULLER P. & PRÉVOT L. (2004). Tableaux conversationnels en sdrt. In *Workshop SDRT, TALN 2004*.

MCTEAR M. F. (2002). Spoken language technology : Enabling the conversational user interface. In *ACM Computer Surveys 34 (1)* : ACM.

MELLISH C., SCOTT D., CAHILL L., PAIVA D., EVANS R. & REAPE M. (2006). A reference architecture for natural language generation systems. In *Journal of Natural Language Engineering 12 (1)*, p. 1–34 : Cambridge University Press.

POPESCU V., CAELEN J. & BURILEANU C. (2007). Generic architecture for natural language generation in spoken human-computer dialogue. In C. BURILEANU, Ed., *The 4th Conference on Speech Technology and Human-Computer Dialogue SpeD 2007* : Romanian Academy Publishing House.

REITER E. & DALE R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

STENT A. (2001). *Dialogue Systems as Conversational Partners : Applying Conversation Acts Theory to Natural Language Generation for Task-Oriented Mixed-Initiative Spoken Dialogue*. Ph D Thesis, University of Rochester.

STONE M. (1998). *Modality in Dialogue : Planning, Pragmatics and Computation*. Ph D Thesis, University of Pennsylvania.

THEUNE M. (2000). *From Data to Speech : Language Generation in Context*. Ph D Thesis, University of Eindhoven.