Un Modèle Cognitif pour la Résolution de la Référence dans le Dialogue Homme-Machine

Sébastien Gérard[†] et Jean-Paul Sansonnet[‡]

LIMSI-CNRS - Université de Paris-Sud
Bâtiment 508 - 91403 Orsay, France
{gerard[†], jps[‡]}@limsi.fr

Résumé – Abstract

Dans cette étude, nous proposons un modèle pour la résolution de la référence dans le cadre du dialogue homme-machine. Partant de considérations psychologiques sur la nécessité d'un partage du système inférenciel pour permettre la communication, nous définissons un formalisme basé sur des règles de production associées à des coûts cognitifs. Au travers d'exemples, nous montrons comment ce formalisme peut être utilisé comme cadre pour intégrer le traitement de différents phénomènes liés à la référence, et comment cette intégration peut conduire à des interfaces en langue naturelle plus efficaces.

In this study, we propose a model for reference resolution in the context of human-computer interaction. Starting from psychological considerations on the necessity of a common system of inferential strategies in order to allow communication, we define a formalism based on production rules associated with cognitive costs. Through examples, we will show how this formalism can be used as a framework to integrate the processing of various reference phenomena, and how this integration can lead to more efficient interactive systems.

Mots - Clefs : Dialogue Homme-Machine, traitement de la référence.

1 Introduction

Comme indiqué par (Bach et Harnish 1979), la communication est possible parce que le locuteur et l'auditeur partagent le même système de stratégies inférencielles, ce qui permet à l'auditeur de déduire les intentions communicatives du locuteur d'après ses dires. Ce point de vue se rapproche de celui de Grice (1957, 1975), qui tente de mettre en évidence les règles que l'on doit suivre pour que le déroulement de la conversation soit correct. Néanmoins, cette théorie laisse ouvertes certaines questions particulièrement cruciales pour une mise en œuvre sur ordinateur - notamment une définition formelle de la notion de pertinence d'une proposition, centrale à cette théorie. Sperber et Wilson (1982, 1986) cherchent à remédier à ces questions en donnant une définition plus précise de la notion de pertinence : une

proposition P sera dite pertinente dans un contexte C si et seulement si P a au moins une implication contextuelle (i.e. non déductible de la seule proposition P). Un locuteur respectant le principe de pertinence doit donc choisir, parmi les énoncés possibles, l'un de ceux qui permettent le plus d'implications contextuelles, dans un contexte le plus petit possible et le plus facilement accessible à l'auditeur. Symétriquement, le destinataire est potentiellement producteur : pour comprendre il reconstruit le système de référence et ses intentions, puis intègre les contenus à son propre système de référence. Culioli (1971, 1985) parle ainsi de co-énonciation plutôt que d'énonciation et de destination.

Dans le cadre de systèmes de dialogue homme-machine (DHM), affin de garantir l'ergonomie des interprétations construites par la machine, c'est à dire leur conformité aux attentes de l'utilisateur, le fonctionnement du système mis en œuvre peut avantageusement tirer profit d'une analogie avec la cognition humaine. Le modèle présenté ici pour le traitement de la référence est basé sur cette hypothèse : si l'on peut trouver une "bonne" représentation du contexte, et des règles d'inférences "aussi proches que possible" de celle d'un locuteur humain, alors l'interprétation correcte d'une intervention est celle avec *le plus d'implications* contextuelles, et avec *le coût cognitif le plus faible*. La difficulté est alors de déterminer dans un premier temps quelles sont ces "bonnes" règles d'inférence et représentations, puis dans un deuxième temps de déterminer les coûts cognitifs. Nous ne prétendons pas avoir atteint cet objectif ; nous nous sommes plutôt attachés à définir un cadre permettant d'intégrer différents aspects du traitement automatique des langues dans un formalisme unique. Pour représenter les règles d'inférence, nous avons choisi d'utiliser des règles de production (Newell, 1990 ; Lugger 1994) car ce formalisme a permis de bons résultats tant pour l'implémentation d'algorithmes de recherche que pour modéliser la manière dont l'homme résout des problèmes. Dans notre modèle, une intervention est une expression que le système va réécrire de toutes les manières possibles en utilisant ces règles et le contexte.

Notre thèse est que pour le traitement de la référence, l'intention - i.e. le fait que le locuteur a choisi une manière particulière d'exprimer quelque chose parmi toutes celles possibles - doit être prise en compte dans le modèle : nous définissons la "bonne" interprétation d'une phrase comme celle dont le coût cognitif est le plus faible. Ceci conduit à notre définition calculatoire du coût cognitif : nous voulons que notre modèle d'accessibilité suive les mêmes principes que pour la cognition humaine. Pour ceci, nous construisons une estimation numérique de l'accessibilité en associant des poids numériques à chaque étape du processus de résolution de la référence. Pour définir le contexte, nous nous appuyons sur la notion de vue définie pour le projet InterViews (Sansonnet, 1999). VDL (View Design Language - Langage de Conception de Vue) est un formalisme en cours de développement destiné à faciliter la conception de médiateurs entre un humain et un composant - logiciel ou matériel.

2 Le processus de résolution de la référence

Considérant qu'un contexte décrit en VDL peut être peut être représenté au moyen de règles de production, nous appelons *contexte d'interprétation* un ensemble fini de couples $(\mathcal{R}, C)$ où $\mathcal{R}$ est une règle de production - que nous appellerons aussi *règle de réécriture*, ou *règle de* transformation - d'une expression en une autre, et $C>0$ son coût. Une expression est un arbre ordonné dont les nœuds sont étiquetés par des symboles, et les feuilles par des symboles ou des mots. Ces symboles peuvent être soit des identifieurs de concept (i.e. l'attribut KEY d'un

concept de la vue), soit des primitives VDL, soit des annotations de référence. Ces annotations sont des informations sur le type de la référence qui proviennent du module de prétraitement ou qui sont produites durant le processus d'interprétation. A l'issue du prétraitement, une intervention de l'utilisateur $\mathcal{S}$ est une expression dont les symboles ne peuvent être ni des identifieurs de concept, ni des primitives VDL. Une interprétation $\mathcal{I}$ d'une expression $\mathcal{S}$ est définie comme une suite finie de règles de transformation de $C\mathcal{I}$. $\mathcal{I}nterpretations(\mathcal{S})$ est alors l'ensemble de toutes les interprétations possibles de $\mathcal{S}$, isomorphe à $\{0, 1, \ldots p\}^{\mathbb{N}}$. Le coût d'une interprétation $\mathcal{I}$ de $\mathcal{S}$, noté $C_S(I)$ est défini comme la somme des coûts élémentaires des transformations de $\mathcal{I}$. L'expression obtenue par l'application successive des règles de transformation de $\mathcal{I}$ sur l'expression initiale $\mathcal{S}$ est appelée résultat de l'interprétation de $\mathcal{I}$ par $\mathcal{S}$. Une interprétation $\mathcal{I}$ de $\mathcal{S}$ est dite correcte si son résultat ne contient que des identifieurs de concepts de la vue ou des primitives VQL (ainsi, l'ensemble du processus d'interprétation peut être vu comme la transformation d'une expression ne contenant que des mots ou des annotations de référence en une expression ne contenant que des identifieurs de concepts ou des primitives VQL). $\mathcal{S}$ est dite compréhensible si elle admet au moins une interprétation correcte.

De manière générale, le problème de déterminer si une intervention non ambiguë de l'utilisateur est compréhensible n'est pas décidable. Nous fixons donc une limite supérieure au coût d'une interprétation. Si aucune interprétation correcte n'est trouvée - l'exploration de l'espace des solutions se fait selon la technique du "branch and bound" (Horowitz and Sahni, 1978) - de coût inférieur à cette limite, alors $\mathcal{S}$ est considérée comme non compréhensible. Réciproquement, dès qu'une interprétation correcte de $\mathcal{S}$ est trouvée, de coût c, alors le système calcule toutes les interprétations de coût inférieur ou égal à c. $\mathcal{M}eilleures(\mathcal{S})$ est inclus dans cet ensemble, et le système peut donc déterminer $Comprehension(\mathcal{S})$. L'implémentation de ce modèle a été faite en utilisant Mathematica. C'est un moteur de réécriture symbolique, donc particulièrement adapté. Il dispose en outre de puissantes fonctions de "pattern matching" ce qui permet une écriture concise des règles de transformation. Dans cette implémentation, nous avons utilisé la syntaxe de Mathematica pour définir les expressions et les règles de transformation. Pour faciliter l'écriture des règles, nous supposons que celles-ci sont locales, c'est à dire qu'elles peuvent s'appliquer à n'importe quel sous-arbre d'une expression.

3 Référence lexicale

Dans le langage VDL, 2 attributs servent à la désignation d'un concept : KEY et LEX. L'attribut KEY, qui sert d'identifieur unique interne, ne peut évidemment pas être supposé connu de l'utilisateur, et n'apparaîtra donc jamais dans ses interactions avec le système. L'attribut LEX sert à faire la liaison entre la langue naturelle et cet identifieur interne. Le concepteur de la vue peut associer un (ou plusieurs) synset(s) de WordNet (Fellbaum, 1998) à chaque concept, définissant ainsi son "sens". Ce que nous appelons référence lexicale est alors le processus permettant de déterminer à quel concept - i.e. à quel identifieur de concept - correspond un lemme présent dans une intervention de l'utilisateur. Fondamentalement, un mot - une forme fléchie - peut être réécrit en une forme de base s'il existe une règle de détachement valide permettant cette transformation. Une forme de base peut alors être réécrite comme chacun des synset l'incluant. Ceci donne le premier ensemble de synset associé à un mot ω :

(1) $S_0(w)=\{a_0,a_1 \ldots a_p\}, p \neq 0$

Un synset peut se réécrire en un identifieur de concept si ce synset est inclu dans l'attribut LEX de ce concept - il est possible que plusieurs concepts "partagent" un même synset ; dans ce cas, le synset pourra être réécrit en chacun des identifieurs des concepts concernés. Ceci permet à des synonymes *stricts* (au sens de WordNet) d'être interprété comme des références au même concept. Pour augmenter la flexibilité de la désignation des concepts, nous utilisons aussi certaines des relations sémantiques définies dans WordNet (hypéronymie/hyponymie pour les noms et les verbes, similarité pour les verbes, adjectifs et adverbes, et méronymie/holonymie pour les noms.). Un synset peut alors aussi se réécrire en un autre synset si ces deux synsets sont liés par une de ces relations. Ceci amène à notre définition récursive de l'enveloppe sémantique $S(w)$ d'un mot :

(2) $S_{n+1}(w)=\bigcup_{a \in S_n(w)} p(a), \qquad S(w)=\bigcup_{n \geq 0} S_n(w)$

où $p(a)$ est l'ensemble des synsets en relation - i.e. l'une des relations sémantiques précédemment décrites - avec a. L'ensemble $\Gamma(w)$ des concepts qui peuvent être référés par un mot w est défini comme l'ensemble de tous les concepts de la vue dont l'attribut LEX contient au moins un des synset de l'enveloppe de w :

(3) $\Gamma(w)=\{c \in Vue \mid Lex(c) \cap S(w) \neq \emptyset\}$

4 Référence aux objets

Pour illustrer comment notre modèle traite la référence aux objets, nous allons prendre l'exemple de la référence à un concept existant de la vue au moyen d'autres concepts. Nous notons REF_CONCEPT[c1, c2, c3…] une telle référence. Nous avons mentionné précédemment que les concepts VDL sont définis par des attributs. Ces attributs sont eux-mêmes définis comme des expressions référant à d'autres concepts. Ainsi, nous construisons à partir du contexte VDL un graphe, le graphe référentiel noté GR, où les nœuds sont étiquetés par les identifieurs des concepts de la vue. Deux concepts c1 et c2 sont alors reliés par un arc (non orienté) si l'un des attributs TYPE, ISA, POF, ACT, INIT, VAL ou SHOW de c1 réfère c2, ou si l'un des mêmes attributs de c2 réfère à c1.

Nous définissons alors la règle de transformation (R1), qui va réécrire tout concept présent dans une annotation de référence `REF_CONCEPT` en l'ensemble des nœuds voisins. Ce mécanisme s'applique récursivement aux ensembles générés. Symétriquement, la règle (R2) réécrit deux ensembles de concepts en leur intersection si celle-ci est non vide. Finalement, la règle de "terminaison" (R3) réécrit `REF_CONCEPT[{c}]` en `c` - dans le cas où l'interprétation a permis de discriminer un concept unique - et la règle (R4) réécrit `REF_CONCEPT[{c1, c2, c3…}]` en `{c1, c2, c3…}` dans le cas où l'ambiguïté n'a pas pu être levée de manière interne (elle a un coût supérieur à (R3)). Ces règles génèrent un nombre relativement important d'interprétations d'une référence `REF_CONCEPT`. Deux considérations atténuent néanmoins la portée de cette remarque : le contexte externe à `REF_CONCEPT` dans une expression en invalidera certaines ; le formalisme VDL utilisé pour décrire un composant crée un cadre référentiel suffisamment contraint pour que les interprétations les plus "naturelles" soient effectivement celles dont le coût est le plus faible.

(R1). $\forall c \in Vue, REF\_CONCEPT[\_\_\_, c, \_\_\_] \Rightarrow REF\_CONCEPT[\_\_\_, \{c_j \in Vue \mid (c, c_j) \in GR\}, \_\_\_]$

(R2). $\forall E_1, E_2 \in P(Vue)\ tq\ E_1 \cdot E_2 \neq \emptyset$,
$REF\_CONCEPT[\_\_\_, E_1, \_\_\_, E_2, \_\_\_] \Rightarrow REF\_CONCEPT[\_\_\_, E_1 \cdot E_2, \_\_\_]$

(R3). $\forall c \in Vue, REF\_CONCEPT[\{c\}] \Rightarrow c$

(R4). $\forall E \in P(Vue), REF\_CONCEPT[E] \Rightarrow E$

5 Référence aux actions

Pour traiter le problème de la référence aux actions, nous utilisons le champ TYPE de chaque concept. Nous définissons la règle de transformation (R5) qui réécrit un concept d'action suivi de concepts dont les types sont compatibles avec ceux des paramètres de l'action (ou toute permutation) en une primitive VQL `APPLY`, qui représente l'application d'un concept d'action sur des paramètres. Ainsi, l'annotation de référence à une action étant `REF_ACTION`, `REF_ACTION[action, paramètre1, paramètre2]` sera transformé en `résultat[APPLY[action, paramètre1, paramètre2]]` si les types de `paramètre1` et `paramètre2` sont corrects. Ici, résultat est un concept qui indique soit le concept résultat de l'action lorsqu'il s'agit d'une action simulable par le médiateur, soit une indication sur son type dans les autres cas.

(R5). $\forall i \neq 1, \forall c_1, c_2, ..., c_i \in Vue\ telsque\ TYPE[c_1] = FUNC[c_2, ..., c_i],$
$\forall (c_1', c_2', ..., c_i') \in Perm(c_1, c_2, ..., c_i), REF\_ACTION[c_1', c_2', ..., c_i'] \Rightarrow APPLY[c_1, c_2, ..., c_i]$

(R6). $\forall c \in Vue, REF\_ACTION[\_\_\_, c, \_\_\_] \Rightarrow REF\_ACTION[\_\_\_, ISA[c][c], \_\_\_]$

(R7). $\forall i \neq 1, \forall c_1, c_2, ..., c_i \in Vue\ telsque\ \$c \in Vue\ et\ TYPE[c] = FUNC[c_1, c_2 ..., c_i],$
$\forall (c_1', c_2', ..., c_i') \in Perm(c_1, c_2, ..., c_i), REF\_ACTION[c_1', c_2', ..., c_i'] \Rightarrow APPLY[c, c_1, c_2, ..., c_i]$

La règle d'héritage **Error! Reference source not found.** utilise l'attribut ISA de chaque concept pour éviter de sur-contraindre les paramètres, réécrivant un concept en lui ajoutant son "père" comme tête. Ainsi, `REF_ACTION[action, paramètre1, paramètre2]` pourra être réécrit en `REF_ACTION[action, type1[paramètre1], paramètre2]` si `ISA[paramètre1]=type1`, aboutissant ensuite à l'application éventuelle de la règle associée à `action`. Pour une plus grande robustesse du traitement de la référence aux actions, nous avons souhaité ajouter la règle **Error! Reference source not found.** traitant les cas où l'action est omise. Cette règle peut réécrire une séquence de paramètres en l'application d'une action de signature compatible sur ces paramètres, sans que le concept associé à l'action ne soit présent. Cette règle a un coût supérieur aux autres, et ne sera donc utilisée que "quand cela est nécessaire".

6 Evaluation et Perspectives

Cette étude montre qu'il est possible d'intégrer différents aspects du traitement de la référence en utilisant un même formalisme basé sur des règles de production sans présupposer de hiérarchie entre les règles. L'introduction de coûts cognitifs associés à des règles génériques et peu contraintes pour le traitement de la référence rend cette intégration possible et efficace,

tout en permettant de prendre en compte l'intention du locuteur. Notre thèse est que ceci est nécessaire pour arriver au but d'un système robuste de DHM. Le principal point faible de cette approche est évidemment que le système doit prendre en compte un nombre important d'interprétations possibles. Le coût cognitif associé à chaque règle doit néanmoins apporter une heuristique lors de l'exploration de ces possibilités. Pourtant, tout au long de cette étude, nous n'avons donné que très peu d'indications sur les coûts associés à chaque règle. Dans l'implémentation actuelle, la majorité des coûts sont fixés à 1, sauf dans le cas de règles écrites pour simuler l'application répétée d'une règle élémentaire, ou dans le cas des règles qui nous semblaient ne devoir être utilisées "en dernier ressort". Il aurait été illusoire de vouloir fixer ces règles de manière *ad hoc*.

Malgré ces limites, les tests effectués sur de petites applications ont montré que l'interprétation des références était correcte dans la plupart des cas. Les interventions analysées étaient limitées à ce qu'il est possible de représenter en utilisant le langage VDL, mais incluaient néanmoins des références complexes, ce qui nous encourage à développer notre modèle. Pour ceci, nous voyons deux axes de travail prioritaires : Le premier est d'améliorer la diversité des interventions que le système peut traiter. Il s'agit d'une part de développer le langage VQL, et d'autre part d'intégrer de nouvelles règles. Le deuxième est évidemment de déterminer les valeurs pertinentes (si ce n'est cognitivement, du moins opérationnellement) des coûts associés à chaque règle.

Références

Bach K. et Hamish R. (1979), *Linguistic Communication and Speech Acts*. MIT Press, Cambridge.

Culioli A. (1985), *Pour une linguistique de l'énonciation,* Ophrys, Paris.

Fellbaum C. (1998), WordNet *: An Electronic Lexical Database*. MIT Press, Cambridge.

Grice P. (1957), Meaning. *Psychological Review, 66,* pp 377-388.

Grice P. (1975), Logic and conversation. *Syntax and Semantics (3) speech acts,* pp 41-58, Academic Press, New York.

Horowitz E. et Sahni S. (1978), *Fundamentals of Computer Algorithms*. Computer Science Press, Rockville.

Luger G.F. (1994), *Cognitive Science : The Science of Intelligent Systems*. Academic Press, San Diego et New York.

Newell A. (1990), *Unified Theories of Cognition*. Harvard University Press, Cambridge.

Norman D. A. (1998), *The invisible computer*. MIT Press, Cambridge.

Sansonnet J.P. (1999), *Description technique du projet InterViews*. Rapport technique n° 99-01, Limsi, Orsay.

Sperber D. et Wilson D. (1982),Mutual Knowledge and relevance in theories of comprehension. *Mutual knowledge*, pp 61-131, Academic press, London.

Sperber D. et Wilson D. (1986), *Relevance, communication and cognition*, Basil Blackwell, Oxford.