Vers une planification basée sur une Théorie Constructive des Types en logique Intuitionniste.

Patrick Barlatier, Richard Dapoigny

LISTIC (EA 3703), Eq. LS
Domaine Universitaire, B.P. 80439, 74944 Annecy-le-Vieux cedex
patrick.barlatier@univ-savoie.fr et
http://www.listic.univ-savoie.fr/
richard.dapoigny@univ-savoie.fr et
http://www.listic.univ-savoie.fr/

Résumé : Ce papier propose une nouvelle sémantique pour la logique de planification dans laquelle on considère un but comme une fonction de son contexte (le contexte étant une structure de connaissance sur le domaine). Étant donnés un but global et une situation initiale, le modèle de planification est généré directement à partir d'un ensemble structuré de buts primitifs par un raisonnement sur les types de buts et la connaissance du domaine. Cette dernière est extraite d'une ontologie locale de domaine par une sélection précise et ordonnée des entités disponibles et de leurs relations. Formellement, cette sélection est modélisée par des types d'enregistrements dépendants de la théorie intuitionniste des types (ITT). Ainsi, en utilisant un démonstrateur de théorème reposant sur cette théorie, il est possible de vérifier la validité (plus précisément le type) d'un ensemble d'objets instanciés représentant la connaissance et l'état actuel du système. Cette identification par un type correspond à une connaissance partielle du domaine associée à un but pour produire une action.

Mots-clés : Intuitionnisme, contraintes, types, sous-typage, planification, représentation des connaissances.

1 Introduction

En planification, il existe à une demande croissante pour une meilleure expressivité des connaissances du domaine afin d'améliorer les performances et de diminuer la complexité des planificateurs (Wilkins & DesJardins, 2001). Le besoin de dépasser les limites imposées par les générateurs de plans automatiques a débouché sur l'intégration de l'ingénierie des connaissances comme nouveau champ d'investigation (Doniat & Aylett, 2001; Barták & McCluskey, 2006). L'acquisition des connaissances sur le domaine et son contrôle peut également améliorer les performances des planificateurs mais nécessite une interaction entre l'utilisateur et le système via des langages contrô-

lés (Ferguson & Allen, 1998; Cortellesa & Cesta, 2006). Ce dernier aspect démontre l'existence d'un lien entre le Traitement Automatique des Langues Naturelles (TALN) et la planification.

Par ailleurs, dans les applications nécessitant un raisonnement sur les connaissances des systèmes, les approches traditionnelles telles que le *Situation Calculus* décrivent l'état du système en utilisant l'hypothèses du monde clos (Etzioni *et al.*, 1995; Reiter, 1978) qui est une notion non axiomatisable (Girard, 2006). De telles hypothèses supposent que l'information sur le monde est complète et qui plus est correcte. En outre, elles induisent de nombreux problèmes tels que le *frame problem*.

Dans le domaine de la planification, nous proposons dans cet article une approche différente pour répondre au problème de l'expressivité en terme de modélisation des connaissances. Ce modèle est basé sur la logique intuitionniste et les types dépendants. Son objectif est d'automatiser des raisonnements corrects sur ces connaissances dans le cadre de la planification de processus physiques. Le modèle est centré sur la représentation de trois concepts, le contexte des actions, les actions ainsi que les effets correspondants incluant le but. Pour cela, nous utilisons un formalisme de représentation intégrant ces concepts dans des structures de types dépendants. Ce formalisme permet d'expliciter la sémantique des diverses composantes des concepts et facilite la planification par la notion de sous-typage, méthode représentant une recherche de *pattern* à grain moins fin que les approches traditionnelles.

2 Bases Logiques

L'hypothèse sous-jacente à la présente approche utilise des connaissances partielles sur un système rassemblées sous le nom de contexte d'un processus physique. S'il n'est pas possible de connaître l'état total d'un système, il est envisageable de vérifier si certaines portions de cet état (contextes) sont valides et d'en déduire l'action possible ainsi que ses effets. Pour cela une logique adaptée associée à un démonstrateur de théorèmes est un élément essentiel. Les dernières décennies ont vu l'émergence de nombreuses logiques et parmi celles-ci, la logique intuitionniste ou logique constructive (Girard, 1989; Coquand & Coquand, 1999; Martin-Löf, 1982; Boldini & Bourdeau, 2004). C'est une logique de la connaissance aussi appelée logique active par opposition à la logique classique ou passive (Huet *et al.*, 1995). En effet, elle stipule que les objets dont les démonstrations intuitionnistes affirment l'existence sont effectivement calculables. Elle permet également, la définition et l'automatisation de raisonnements corrects sur des informations issues d'une base de données ou d'une ontologie. Seules les observations fournissent des informations, et l'absence d'observation d'une propriété ne peut être utilisée pour prouver que celle-ci est fausse. Les seuls éléments qui entrent en compte dans la représentation des croyances sont d'une part la connaissance de la structure du système (ontologie des descriptions possibles), et d'autre part une notion d'ordre partiel sur ces descriptions permettant de les comparer suivant la quantité d'informations qu'elles fournissent.

En observant les objets physiques comme des éléments essentiels de processus physiques et en se basant sur les résultats de travaux récents dans le domaine du traitement des langues naturelles (Boldini, 2000; Ginzburg, 2005; Cooper, 2005; Ranta, 2004),

dans le domaine de la certification de programmes (Bove & Capretta, 2001; Coquand & Coquand, 1999; Paulson, 1989) et plus récemment dans le domaine du Web sémantique (Halpin & Thompson, 2006), nous avons proposé un modèle utilisant la théorie intuitionniste des types (Barlatier & Dapoigny, 2007; Dapoigny & Barlatier, 2006, 2007b) pour modéliser les contextes, les actions et les effets. La représentation des contextes de processus est décrite par des enregistrements à types dépendants. Nous considérons plus particulièrement le *contexte-de* l'action, c'est à dire le *contexte-de* l'action menant au but désiré. Par ailleurs, des travaux ont montré que la représentation de connaissances liées au contexte nécessite une ontologie (Strang & Linnhoff-Popien, 2004). La description des concepts manipulés est fournie par une ontologie de domaine dont les objets serviront de preuves formelles. L'association d'une ontologie au présent modèle formalise et généralise la notion de type.

La théorie intuitionniste des types, que l'on dénotera par ITT par la suite, est issue de la synthèse de la logique intuitionniste et d'un système formel de types. Celle-ci exploite simultanément la logique intuitionniste et la théorie des types du λ-calcul typé en fournissant un système de typage élaboré dans lequel toute propriété en logique des prédicats peut être interprétée comme un type. C'est la base d'importants travaux en logique ainsi qu'en IA (démonstrateurs de théorème tels que Coq, LEGO, ...), languages fonctionnels et logiques (Logique Linéaire, Théorie des types de Martin Löf, Théorie Constructive des types de Coquand, ...). En logique intuitionniste, le sens de chaque constante logique doit être extrait de la spécification de ce qui peut constituer une preuve pour toute proposition dont cette constante est le principal connecteur. Un des apports majeurs de la théorie des types est le paradigme "proofs-as-programs" plus connu sous le nom d'isomorphisme de Curry-Howard associant une preuve constructive à un programme qui réalise la formule prouvée (Howard, 1980). Un autre atout réside dans la calculabité de n'importe quel jugement (Coquand & Coquand, 1999; Martin-Löf, 1982; Valentini, 1996) : la théorie intuitioniste des types est fonctionnellement décidable (Valentini, 1996). Dans le jugement de type $a : T$, on classifie un objet a comme étant de type T. Une innovation de ITT est l'introduction des types dépendants et des enregistrements à types dépendants (Betarte, 2000; Kopylov, 2003). Ceux-ci sont exprimés en termes de données et peuvent exprimer toute connaissance issue d'information en étant plus flexible que les systèmes de types conventionnels (i.e., ils offrent un continuum de précision s'étendant d'une simple assertion de base jusqu'à la spécification complète d'un problème). Le concept de contexte peut être exprimé par des enregistrements à types dépendants intégrant simultanément de simple types, des types de propositions et/ou des types de fonctions. Les types de contexte (ce qui est possible) se distinguent des objets de contexte appelés tokens (la réalité). Cette subdivision types/objets bénéficie en outre du support de la représentation ontologique pour la spécification des applications lors de la conception. La capacité de fournir une structure simple qui peut être ré-utilisée afin de spécifier différentes sortes d'objets sémantiques structurés est un élément clé du modèle (Dapoigny & Barlatier, 2007a).

Definition 1
Un enregistrement à types dépendants est une suite de champs dans laquelle les labels l_i correspondent à certains types T_i, c'est à dire, que chaque champ successif peut

dépendre des valeurs des champs précédents :

$$C = \left[\begin{array}{ll} l_1 & : T_1 \\ l_2 & : T_2(l_1) \\ \ldots & \\ l_n & : T_n(l_1 \ldots l_{n-1}) \end{array}\right] \qquad exemple : C_1 = \left[\begin{array}{ll} x & : Person \\ y & : Ticket \\ p_1 & : own(x, y) \\ z & : Flight \\ p_2 & : has\_Flight(y, z) \\ t & : Town \\ p_3 & : has\_Destination(z, t) \end{array}\right]$$

Une définition similaire introduit les objets (tokens) dans laquelle une suite de valeurs est telle qu'une valeur v_i peut dépendre des valeurs des champs précédents $l_1, ..., l_{i-1}$:

$$c = \left[\begin{array}{ll} l_1 & = v_1 \\ l_2 & = v_2 \\ \ldots & \\ l_n & = v_n \\ \ldots & \end{array}\right] \qquad exemple : c_1 = \left[\begin{array}{ll} x & = John \\ y & = t0015JK \\ p_1 & = q_1 \\ z & = ZU515 \\ p_2 & = q_2 \\ t & = Zurich \\ p_3 & = q_3 \\ \ldots & \end{array}\right]$$

où q_1 est une preuve de $own(John, t0015JK)$, q_2 une preuve que $has\_Flight(John, ZU515)$ et q_3, une preuve de $has\_Destination(John, Zurich)$. Nous allons montrer dans les deux paragraphes suivants comment les enregistrements à types dépendants peuvent représenter les contextes ainsi que la structure intentionnelle associée aux actions et aux buts résultants.

3 Structures de types dépendants associées aux contextes

Dans le cadre des processus physiques, la notion de Context Record Type (CRT) est exprimée par un type d'enregistrements dépendants dans lequel les champs détaillent la connaissance (i.e., les concepts, les propriétés et les contraintes). Il n'y a pas de limite supérieure au nombre de champs. Le CRT vide $<>$ n'impose aucune propriété et aucune contrainte. En supposant que Γ est un contexte valide[1], les règles suivantes introduisent les CRTs comme des enregistrements dépendants ($record - type$) :

$$\frac{}{\Gamma \vdash <> : record - type} \tag{1}$$

$$\frac{\Gamma \vdash R : record - type \;\; \Gamma \vdash T : record - type \rightarrow type}{\Gamma \vdash < R, l : T > : record - type} \tag{2}$$

La première règle affirme l'existence d'un tout, le CRT vide tandis que la seconde définit la formation des enregistrements à types dépendants pourvu que le champ l

[1]Un contexte valide en théorie des types est une suite $x_1 : T_1, \ldots x_n : T_n$ telle qu'il existe un jugement dont la partie gauche d'un séquent comporte cette séquence.

ne soit pas déjà déclaré dans R. Nous supposerons également que les constructions syntaxiques primitives (i.e., égalité, application fonctionnelle et lambda abstraction) sont valables (pour plus de détails voir (Martin-Löf, 1982)).

Un des aspects les plus importants des CRT est la notion de sous-typage. Par exemple, un objet d'enregistrement à types dépendants avec des champs additionnels non mentionnés dans le type de départ est encore du même type. Le mécanisme d'enrichissement d'informations correspond à l'extension d'un type de contexte C par un type de contexte C'. En théorie des types, la proposition et le jugement sont équivalents. Il est alors possible de conclure que le jugement C est étendu (*lifted*) au jugement C'. Comme la résolution du sous-typage nécessite la connaissance de toutes les coercions possibles pour un terme donné ainsi que leurs effets, elle est incalculable en pratique. Nous contournons ce problème en imposant des contraintes sémantiques sur les coercions (Betarte, 2000).

Definition 2
Soient deux CRT C et C', si C contient au moins tous les labels déclarés dans C' et si les types des labels communs sont dans la relation d'inclusion, alors C est un sous-type de C' :

$$C \sqsubseteq C' \tag{3}$$

L'inclusion de types et les règles correspondantes généralisent l'inclusion des CRT aux enregistrements à types dépendants et la propagent aux autres types du language.

4 Structures de types dépendants associées aux buts

Dans la suite, nous supposerons que les majuscules dénotent des types et les minuscules, les objets (ou tokens). L'observation des situations courantes conduit aux hypothèses suivantes :

– Une action peut s'exécuter dans plusieurs contextes.
– Un même contexte ne peut pas être commun à plusieurs actions simultanément.

L'idée de base consiste à considérer le contexte, l'action et les effets (incluant le but) comme des DRTs. On peut alors associer par une fonction l'action au contexte, puis à associer le résultat obtenu aux effets. La théorie des types d'enregistrement dépendants permet pour cela la définition de fonctions et de types de fonctions grâce à une version du λ-calcul typé.

Definition 3
Etant donné un CRT C, une action peut être intuitivement décrite par une famille de types d'enregistrements qui est une fonction d'enregistrements de type CRT vers des types d'enregistrements, comme la λ-abstraction :

$$\lambda c : C.[a : action\_verb(\ldots, c.l_i, \ldots) \tag{4}$$

où $action\_verb$ est un verbe d'action, l_i un champ de c et a une proposition preuve de l'action effectuée.

Les types sont extraits d'une ontologie locale pour composer le CRT servant d'entrée au concept d'action. En utilisant l'exemple précédent, une fonction du type :

$$\lambda c_1 : C_1 . [a : take\_flight(c_1.x, c_1.z)$$

applique un enregistrement de la forme c_1 vers un enregistrement de la forme $[a = r_1$ où r_1 est une preuve de $take\_flight(John, ZU515)$. Les types de base génèrent un contexte dans lequel les types et objets respectifs sont valides (récursivité possible). La notion de contrainte est représentée par une fonction entre DRT (Cooper, 2005). L'action est en fait considérée simultanément comme une conséquence de la présence d'un contexte et comme une pré-condition à l'existence des effets de cette action. Cet aspect correspond à la notion de point fixe, notion qui a notamment permis de décrire la quantification dynamique en TALN. Si $\mathcal{F}$ représente une famille de types d'enregistrement dépendants alors a est un point fixe pour $\mathcal{F}$ si $a : \mathcal{F}(a)$. Supposons que A soit la famille :

$$\lambda c_1 : \left[\begin{array}{ll} x & : Person \\ y & : Ticket \\ p_1 & : own(x, y) \\ z & : Flight \\ p_2 & : has\_Flight(y, z) \\ t & : Town \\ p_3 & : has\_Destination(z, t) \end{array}\right] . [a : take\_flight(c_1.x, c_1.z)$$

alors :

$$\left[\begin{array}{ll} x & : Person \\ y & : Ticket \\ p_1 & : own(x, y) \\ z & : Flight \\ p_2 & : has\_Flight(y, z) \\ t & : Town \\ p_3 & : has\_Destination(z, t) \\ a & : take\_flight(x, z) \end{array}\right]$$

représente le type de tous les points fixes de $\mathcal{A}$. La notion de point fixe est utilisée pour étendre la définition précédente à une structure intentionnelle en considérant le but et ses effets comme une fonction d'un contexte possible. Cette structure appelée OC est définie par une fonction admettant comme argument un enregistrement de type point fixe et qui retourne un type d'enregistrement décrivant le but associé ainsi que d'éventuelles propositions résultant de l'action sur l'environnement[2] :

$$\lambda r : \mathcal{F}(\mathcal{A}). \left[\begin{array}{ll} g : [g_1 : ... & \\ e : \left[\begin{array}{ll} e_1 : & \ldots \\ e_2 : & \ldots \end{array}\right. & \end{array}\right. \quad (5)$$

Cette définition correspond à un lien dans lequel un agent observant une action de type $[a : action\_verb(...)$ prédit l'existence d'un but de type $\left[\ g_1 : \ \ldots\right.$ et des propositions associées. Ce type de fonction exprime le fait que le contexte de sortie (OC) est

[2]Le but est considéré comme un effet particulier.

une conséquence de l'existence d'un objet de type contexte (noté IC) associé à une action. A partir d'un même contexte, une seule action est possible (contexte de l'action). Les contextes résultants (OC) doivent tous différer pour préserver la nature fonctionnelle du modèle (cf fig.1). Cela revient à considérer que chaque contexte est relié à une seule action pour générer un but : l'existence de ce contexte 'cause' l'existence des propositions de l'OC (incluant le but). Notons qu'une action à effets conditionnels ayant des effets qui varient selon le contexte d'application sont bien décrites par ce modèle.

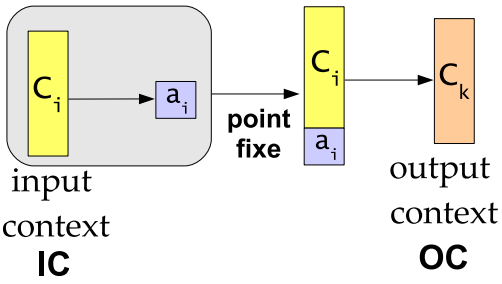

FIG. 1 – Contextes et structures intentionnelles.

5 Application au Planning

L'algorithme de planification est axé sur la causalité qui existe entre les CRTs de type IC et les types OC associés (voir fig. 2).

Proposition 1
Soit un objet valide $c_i : C_i$, s'il existe un objet $c_j : C_j$ tel que :

$$c_j \sqsubseteq \left[\begin{array}{ll} g : [g_1 : ... \\ e : \left[\begin{array}{ll} e_1 : & \ldots \\ e_2 : & \ldots \end{array}\right. \end{array}\right.$$

alors c_i précède c_j.

La planification est réalisée par une progression dans laquelle la direction de recherche part de la situation initiale, mais au lieu de rechercher les états possibles, on recherche les objets de CRT valides et on essaie d'établir des chemins dans lesquels les noeuds sont des CRT via la relation de causalité. L'algorithme nécessite deux parties, une phase de vérification de type pour l'identification des contextes et une phase de backtraking. Le problème de planification peut se réduire au tuple $P = (S, \Sigma, \mathcal{C}, g)$, où S est la liste des contraintes initiales dynamiques (dont les valeurs peuvent changer durant le processus), Σ, les contraintes statiques (au moins pendant la durée du processus), $\mathcal{C}$ dénote l'ensemble des CRT et g le but final à réaliser. Alors $\Pi(S, \Sigma, \mathcal{C}, g)$ représente l'ensemble des plans possibles. La figure 3 dans laquelle a représente une

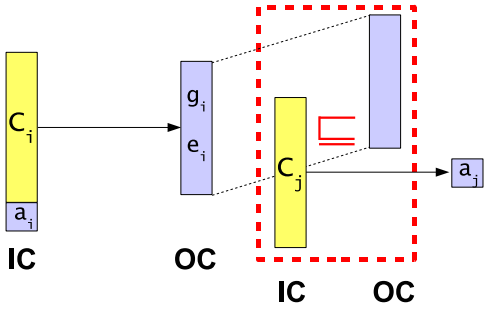

FIG. 2 – Opérateur de chaînage (SEQ).

		Sémantique STRIPS	Sémantique basée sur ITT
Comparaison structurelle	Approche	Epistémologique	Ontologique
	Logique	Premier ordre	ITT
	Hypothèse du monde clos	Oui	Non
	Problème traité	Etat initial + buts + ensemble d'actions	Situation initiale + buts + paires CRT-but
	Grain	Prédicats	DRTs
	Représentation des actions	Préconditions + effets	DRT + fonctions
	Effets conditionnels	Non	Oui
Comparaison fonctionnelle	Mécanisme	- si precond(a)=TRUE dans l'état s alors activer a	- déterminer la situation initiale et le(s) contexte(s) valide(s) par type-checking
		- substitution de variables	-sélection de l'action a
		- détermination de l'état final s' par mise à jour de s avec les effets	-recherche du nouveau contexte par sous-typage
		- arrêt quand le but appartient à s'	- arrêt quand le but est un sous-type du contexte obtenu
	Traitement du frame probleme	Tous les prédicats de s absents des effets restent inchangés	Seules les propositions et les types prouvés sont valides à un instant donné

FIG. 3 – Comparaison avec STRIPS.

action et s, s' des états, résume les différences essentielles avec la référence STRIPS. Au niveau structurel, l'approche basée sur ITT repose sur un modèle ontologique, pré-

sente un grain moins fin et gère les effets conditionnels. Sur le plan fonctionnel, son mécanisme n'utilise que le type-checking et le sous-typage. Enfin, notons que le frame problème est résolu par la logique elle-même qui ne considère que des types prouvés à un instant donné.

6 Conclusion

Le formalisme proposé ne prétend pas résoudre tous les problèmes de sémantique concernant la planification. Il s'agit plutôt, comme le suggère Girard, de proposer un formalisme capable d'unifier des branches aussi différentes de l'IA que le sont le TALN, la planification et la modélisation des web services. En effet, notre formalisme permet de représenter les connaissances en utilisant la théorie intuitionniste des types comme vecteur d'inter-opérabilité. Plus précisément, nous nous distinguons d'une approche classique basée sur une logique du premier ordre, par la possibilité de représenter des informations partielles et l'aspect dynamique de situations, tout en conservant l'avantage d'utiliser une logique décidable pour prouver nos programmes. Enfin, la compatibilité avec les systèmes de traitement des langues naturelles est un atout majeur pour la réalisation d'une interface utilisateur. Nos recherches s'orientent vers la génération d'une interface utilisateur graphique afin de résoudre des problèmes complexes de planification (Ferguson & Allen, 2005) et vers une extension du modèle aux systèmes distribués.

Références

BARLATIER P. & DAPOIGNY R. (2007). Using contexts to prove and share situations. In *Procs. of the 20th international FLAIRS conference*, p. 448–453 : AAAI Press.

BARTÁK R. & MCCLUSKEY L. (2006). The first competition on knowledge engineering for planning and scheduling. *AI magazine*, **27**(1), 97–98.

BETARTE G. (2000). Type checking dependent (record) types and subtyping. *Journal of Functional and Logic Programming*, **10**(2), 137–166.

BOLDINI P. (2000). Formalizing context in intuitionistic type theory. *Fundamenta Informaticae*, **42**, 1–23.

BOLDINI P. & BOURDEAU M. (2004). La theorie constructive des types. *Mathématiques et Sciences humaines*, **165**, 5–12.

BOVE A. & CAPRETTA V. (2001). Nested general recursion and partiality in type theory. In R. BOULTON & P. JACKSON, Eds., *TPHOL*, number 2152 in LNCS, p. 121–135 : Springer.

COOPER R. (2005). Records and record types in semantic theory. *J. Log. Comput.*, **15**(2), 99–112.

COQUAND C. & COQUAND T. (1999). Structured type theory. In *Workshop on Logical Frameworks and Meta-languages*.

CORTELLESA G. & CESTA A. (2006). Feature evaluation in mixed-initiative systems : An experimental approach. In S. S. D. B. DEREK LONG & L. MCCLUSKEY, Eds., *Procs. of ICAPS'06* : AAAI Press.

DAPOIGNY R. & BARLATIER P. (2006). Dependent record types for dynamic context representation. In F. C. M. BRAMER & A. TUSON, Eds., *Research and Development in Intelligent Systems : Procs. of AI-2006*, volume 23 : Springer.

DAPOIGNY R. & BARLATIER P. (2007a). Goal reasoning with context record types. In *Procs. of CONTEXT'07* : Springer. Accepted for publication.

DAPOIGNY R. & BARLATIER P. (2007b). Towards a context theory for context-aware systems. In *Procs. of the 2nd IJCAI Workshop on Artificial Intelligence Techniques for Ambient Intelligence*.

DONIAT C. & AYLETT R. (2001). Planform-ka tool : A cluster of constraint for knowledge acquisition and planning. In *Procs. of the 20th Workshop of the UK Planning and Scheduling Special Interest Group*.

ETZIONI O., GOLDEN K. & WELD D. S. (1995). *Sound and Efficient Closed-World Reasoning for Planning*. Rapport interne TR-95-02-02.

FERGUSON G. & ALLEN J. (1998). Trips : An intelligent integrated problem-solving assistant. In *Procs. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, p. 567–573.

FERGUSON G. & ALLEN J. (2005). Mixed-initiative dialogue systems for collaborative problem-solving. In *Procs. of the AAAI Fall Symposium on Mixed-Initiative Problem Solving Assistants*, p. 57–62.

GINZBURG J. (2005). Abstraction and ontology : Questions as propositional abstracts in type theory with records. *Journal of Log. Comput.*, **15**(2), 113–130.

GIRARD J.-Y. (1989). *Proofs and Types*. Cambridge University Press.

GIRARD J.-Y. (2006). *Le point aveugle*, volume 1 of *Vision des sciences*. Hermann.

HALPIN H. & THOMPSON H. (2006). One document to bind them : combining xml, web services and the semantic web. In *Procs. of the World Wide Web Conference*.

HOWARD W. A. (1980). *To H.B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*, chapter The formulae-as-types notion of construction, p. 479–490. Academic Press.

HUET G., KAHN G. & PAULIN-MOHRING C. (1995). *The Coq Proof Assistant - A tutorial*. Rapport interne 178, INRIA.

KOPYLOV A. (2003). Dependent intersection : A new way of defining records in type theory. In *Procs. of the 18th IEEE Symposium on Logic in Computer Science*, p. 86–95.

MARTIN-LÖF P. (1982). Constructive mathematics and computer programming. *Logic, Methodology and Philosophy of Sciences*, **6**, 153–175.

PAULSON L. (1989). The foundation of a generic theorem prover. *Journal of Automated Reasoning*, **5**, 363–397.

RANTA A. (2004). Grammatical framework : A type-theoretical grammar formalism. *Journal of Functional Programming*, **14**(2), 145–189.

REITER R. (1978). On closed world databases. In H. GALLAIRE & J. MINKER, Eds., *Proc. of ACM SIGMOD Int. Conf. on Management of Data*.

STRANG T. & LINNHOFF-POPIEN C. (2004). A context modeling survey. In *Sixth International Conference on Ubiquitous Computing (UbiComp2004)*, p. 34–41.

VALENTINI S. (1996). Decidability in intuitionistic type theory is functionally decidable. *Mathematical Logic*, **42**, 300–304.

WILKINS D. & DESJARDINS M. (2001). A call for knowledge-based planning. *AI Magazine*, **22**(1), 99–115.