Causal Reasoning with Contexts using Dependent Types

Richard Dapoigny and **Patrick Barlatier**
LISTIC/Polytech'Savoie, University of Savoie, France,
email:richard.dapoigny@univ-savoie.fr

Abstract

In Artificial Intelligence, a crucial requirement is the ability to reason about actions and their effects on the environment. Traditional approaches which rely on classical logic suffer from a number of limitations such as semi-decidability, closed-world assumption, frame problem, and a lack of ability to cope with partial knowledge and dynamic environments. While many solutions have been proposed for these topics, difficulties and uncertainties remain, even in the latest papers. In this paper, we look at this issue from a proof-theoretical perspective and we describe the foundations for reasoning about action based on Constructive Type Theory.

Introduction

A crucial requirement for intelligent systems is the ability to reason about actions and effects on the world while avoiding standard problems of classical logic. The basic idea is to consider a theory in which knowledge is incoming constructively (as constructive proofs) from observations. It is axiomatized entirely in a fragment of Constructive Type Theory (CTT). The logic employs the so-called open-world assumption, which means that if something is not explicitly said to be false, it is not assumed to be false (the knowledge about the world is assumed to be incomplete). The system includes an ontology which describes *entities* as the first component of knowledge and *proposition* as the second component, provided that knowledge consists in having evidence for a proposition. The present approach is causal, highly dynamic with information structures while preserving the algorithmic complexity.

The Action-based Theory

Data Structures

The basic idea of the present work is to apply the formalism of CTT into an action-based logical process in which it is able to serve as a background for expressing knowledge via Dependent Record Types (DRTs). The two basic dependent types are the Π-types and the Σ-types. Π-types model functions whose output type may vary according to the input (i.e., dependent functions). As a consequence, predicates can be interpreted as functions from some object to a proposition. For instance, one may define the following Π-type in order to represent the fact that a lift referred as $Sch0437$ has a door: $has\_door : (\Pi x : lift.P(x))$ in which $P(x)$ stands for a proposition that depends on x. An instance of the Π-type would be $has\_door(Sch0437) : P(x)$. Similarly, Σ-types model pairs in which the second component depends on the first. Let us consider the following proposition $boarded(x)$ such as $boarded : (\Pi x : passenger.P(x))$. If q is an object having the type, $q : (\Pi x : passenger.boarded(x))$ then one can define the Σ-type $(\Sigma x : passenger.boarded(x))$ whose object $< peter, q(peter) >$ indicates that for the individual $peter$, the proposition is proved. DRTs (Betarte 2000)(Kopylov 2003) are an extension of Π-types and Σ-types in which types are expressed in terms of data. Dependent record types are much more flexible than simple dependent types (McKinna 2006). They realize a continuum of precision from the basic assertions we are used to expect from types, up to a complete specification of a representation (e.g., a context).

Definition 1 *A dependent record type is a sequence of fields in which labels l_i correspond to certain types T_i, that is, each successive field type can depend on the values of the preceding fields:*

$$< l_1 : T_1, l_2 : T_2(l_1) \ldots, l_n : T_n(l_1 \ldots l_{n-1}) > \qquad (1)$$

where the type T_i may depend on the preceding labels $l_1, ..., l_{i-1}$.

The fields of the Context Dependent Record Type (C-DRT) detail the domain knowledge (i.e., concepts, their properties and their constraints). Their simple structure can be reused to specify different kinds of structured semantic objects (Dapoigny and Barlatier 2007).

$$\underbrace{\begin{bmatrix} l_1 & : lift \\ x_1 & : passenger \\ p_1 & : boarded(x_1) \\ f_1 & : floor \\ p_2 & : destination(x_1, f_1) \end{bmatrix}}_{C_1 : Context\ type} \qquad \underbrace{\begin{bmatrix} \ldots & \\ l_1 & = Sch0437 \\ x_1 & = peter \\ p_1 & = q_1 \\ f_1 & = 5 \\ p_2 & = q_2 \\ \ldots & \end{bmatrix}}_{c_1 : C_1}$$

in which q_1 and q_2 are respective proofs of $boarded(peter)$ and $destination(peter, 5)$. Dots in the context token express that any information can be added in the record token provided that it does not contradict the previous one (partial knowledge). The resulting token is still of the type C_1. An important aspect of DRT is that sub-typing is allowed.

Definition 2 *Given two record types C and C', if C' contains at least every label declared in C and if the types of these common labels are in the inclusion relation then C is a subtype of C' which is written:*

$$C \sqsubseteq C' \tag{2}$$

Semantics

The action-based theory automates a succession of actions to determine, which of them lead to the desired effects. An action can be implemented, if its related context is proved. After its execution, the action has some effects by which the environment changes. For that purpose, an automaton searches among the new contexts, selects the appropriate one among the available ones, executes its related action and loops again until the final goal is obtained (a particular effect). In this constructive theory, the (partial) state of the world is expressed in terms of dependent record types relativized to a particular situation. Given a situation $\mathcal{S}$, the following axioms hold:

A1 A situation $\mathcal{S}$ is described by a set of objects (tokens) corresponding to basic types and propositions types.

A2 A DRT holds in $\mathcal{S}$ iff all its fields are proved.

A3 A finite number of C-DRT (possibly empty) can be valid in $\mathcal{S}$.

A4 A situation is consistent if it is non empty.

A5 A situation is complete if it contains at least a DRT token.

A6 A given action is related at least with one C-DRT.

A7 A given C-DRT cannot be related to several actions.

The action a maps a context type C-DRT into an effect type (E-DRT). The theory induces a tree of context types rooted at the empty record type (default type). This approach is similar in spirit with that of (Thielscher 1997), in which the domain knowledge is described by a triple (C, A, E), but here it is expressed as a family of DRTs $< c : C - DRT, a : A - DRT(c), e : E - DRT(a, c) >$.

Definition 3 *Let C a context record type, A, an action proposition type not occurring in C and E an effect record type, an action specification is described by the DRT $\sigma :< c : C, a : A(c), e : E(c, a) >$.*

Definition 4 *If S is a record of type $\mathcal{S}$ describing the current situation, we say that the specification σ is available in S iff $\mathcal{S} \sqsubseteq C$, i.e., the types occurring in the situation must form a subtype of the context record type.*

The causality operates from context-action pairs to $E - DRT$. Let us consider the context C_1 defined in subsection 1. The action $stops(l_1, f_1)$ provides the (minimal) effect: $e_1 : served(x_1)$ and the action specification is written: $< c_1 : C_1, a : stops(c_1.l_1, c_1.f_1), e : served(c_1.x_1) >$, while a proof of this action specification is $< c_1, stops(Sch0437, 5), served(peter) >$.

The expressivity of the part $< c : C, a : A(c) >$ in σ includes multiple conditions within a single framework. Also, the formalism accounts for actions with conditional effects. Let consider two action specifications (for the same action), $\sigma_1 = (c_1 : C_1, a : A(c_1), e_1 : E_1(c_1, a_1))$ and $\sigma_2 = (c_2 : C_2, a : A(c_2), e_2 : E_2(c_2, a_2))$. It is trivial to see that the C-DRT c_1 related to the action a causes the effect(s) in e_1, whereas the occurrence of a context of type c_2 change the effect(s) to e_2. This situation can be easily extended to any number of $C - DRT$. A causal relationship exists only if the C-DRT and its related action exist. In summary, the same action put in different contexts produces different effects: this formalizes the concept of conditional effects.

Conclusion

DRTs are able to pack up data structures, operations over them, and also proofs of the properties of those operations. To some extent, the notions of C-DRT, actions and E-DRT correspond to the well-known STRIPS triple pre-conditions, action and post-conditions. However, the framework in which they appear here is radically different. Since we construct proofs after each occurrence of a new situation, and above the assumption of causal reasoning, the frame problem no longer exists. The ramification problem doesn't occur since action specifications are not assumed to completely describe all the possible effects but only a minimal collection of effects. Making the information more precise with DRTs, the theory is more expressive than existing ones. This great expressive power together with the close relation to ontologies are a central element in areas such as context-aware applications or web services. The present action theory also supports both nondeterministic choice of actions and the specification of actions that have nondeterministic effects[1]. However, the price to pay for such benefits is that it requires the specification of appropriate context types which is known to be a difficult task.

References

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

Dapoigny, R., and Barlatier, P. 2007. Towards a context theory for context-aware systems. In *Procs. of the 2nd IJCAI Workshop on Artificial Intelligence Techniques for Ambient Intelligence*.

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*, 86–95.

McKinna, J. 2006. Why dependent types matter. *SIGPLAN* 41(1).

Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence* 89(1-2):317–364.

[1]since any E-DRT may result in an infinite number of objects of this type.