PPDR: A Proof Protocol for Deductive Reasoning

Patrick J. Hayes[1] Paulo Pinheiro da Silva[2]
Deborah L. McGuinness[2] Richard Fikes[2]

[1]Institute for Human and Machine Cognition
Pensacola, FL 32502
e-mail: phayes@ihmc.us
[2]Knowledge Systems Laboratory, Stanford University
Stanford, CA 94305, USA.
e-mail: {pp,dlm,fikes}@ksl.stanford.edu

Abstract

1 Introduction

We need to be able to state rules in terms of generic forms or patterns over SCL [] (or KIF [1]) rather than SCL itself.

We need to describe why we are using SCL and what is the relationship between SCL and the Semantic Web

The work described this paper is developed in the context of the Proof Markup Language (PML) [3], which is an interlingua for capturing the information agents need to understand results and to justify why they should believe the results. *NodeSet* and *InferenceStep* are the main PML constructs for proofs and explanations. A NodeSet represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with the NodeSet. An InferenceStep represents a justification for the conclusion of a node set. Many are the syntactic conditions for a set of PML document representing proofs to be well-formed. An important syntactic conditions for having a well-formed set of PML documents, however, is not in PML itself: inference steps are correct application of rules on inference step premises.

This paper introduces PPDR – A Proof Protocol for Deductive Reasoning, which is a language for stating rules in terms of SCL patterns. Thus, a PML proof is said to be PPDR compliant if we can observe the following in it:

- node set conclusions are either written in SCL or they can be correctly, completely translated from their original languages into SCL;

- inference step rules are specified in PPDR and the application of the rules can be verified by matching PPDR specifications against inference step premises and conclusion.

Many are the benefits of PPDR compliant PML documents including the ability of performing the following:

- verifying if inference step are correct applications of inference rules;
- combining proofs on hybrid reasoning contexts;
- abstracting proofs by matching and replacing rules by more abstract derived rules (explanation generation from proofs).

The rest of the paper is organized as follows. Section 2 introduces the PPDR language specification. Section 3 describes how rules registered on the Inference Web Base [2] are specified in PPDR and how these rule specifications can be used to check PML proofs.

2 The Proof Protocol for Deductive Reasoning

2.1 SCL Schemas

Say that a schema is any expression of SCL in which some pieces of a certain grammatical category (typically things like Sent(ence), Var(iable), Rel(ation symbol) etc.) have been replaced by a schematic variable (or meta-variable), with the corresponding type noted. So, as SCL schema has the general form

pattern ;; *syntax-conditions*

where

- the *pattern* is an SCL expression with schematic variables and substitutions. The % sign is a schematic variable prefix on schema patterns and expression pieces replaced by schematic variables are *placeholders*. A schema pattern must have at least one placeholder. Patterns use $p[s/t]$ as a substitution meaning p with t substituted for s.
- the *syntax-conditions* are specialized SCL expressions specifying the types of the schematic variables.

(implies %p %q);; (Sent %p %q)

is a schema for any SCL implication sentence. There, %p and %q are placeholders for SCL sentences and their replacing variables must be SCL sentences.

2.2 List Notation

A notation for representing lists of schematic variables is required for SCL schemas that can have a variable number of schematic variables of the same type. Definitions for list ranges and lists are presented as follows:

- A *list range* has a *left* and a *right* boundaries connected by an ellipsis, e.g., $1...m$. Range boundaries can be either natural numbers or natural number variables. The list range represents the sequence of natural numbers from the left boundary to the right boundary, including both boundaries. If i is the left boundary and j is the right boundary of an range and $i > j$ then the range is said to be "empty".

- A *primitive list* of schematic variables is represented by a pair of curly brackets containing two arguments: a label and a list range (e.g., { *label*, *range* }). The list arguments are defined as follows:

  - The *label* is the name of the list; Once a list is defined, it can be referred as $\{label\}$. For instance, a list can be defined in the pattern part of a schema and referred in the syntax-conditions part of the schema.

- A *derived list* of schematic variables is represented by a pair of curly brackets containing three arguments: a label, the label of a pre-defined list, and a list of list elements (e.g., $\{label, from-list-label, [operation]list-element*\}$). The arguments are defined as follows.

  - The *from-list-label* identifies the list from where the current list is derived. For example, $\{M, L, -\%x\}$ is "derived" from $\{L\}$;
  - The *[operation]list-element** argument is a list of elements related to derived lists. Each element is proceeded by an operation, which defines the relationship between the element and the derived list. The list of elements can have one or more elements, each one proceeded by an operator. Currently just the "-" operation is specified and it means that for lists $\{L, i...j\}$ and $\{M, L, -\%p\}$ $\%p$ is an element of $\{L\}$ that was not added to $\{M\}$ during the derivation process. One can think $\{M\}$ as the result of removing $\%p$ from $\{L\}$. Thus, for example, if L is the set of elements in $\{L\}$ and M is the set of elements in $\{M\}$ then $M = L - \{\%p\}$.

By using the list notation, we can specify new kinds of SCL schemas. For example,

(and {N,1...k}) ;; (Sent {N})

is a schema for any SCL conjunction sentence. The example above also shows how the type of list elements are specified in syntactic conditions.

2.3 Rule Schema

In PPDR compliant PML documents, a proof is a structure (not quite a tree) generated by rules which have the general form

> *name*: *premises* |− *conclusion* ;; *syntax-conditions* ;; *discharged-sentences*

where premises is a list of sentence schemas and conclusion is a sentence schema. A proof schema is just like a rule with sentence-schema antecedents. The following example

> ndUI: (forall ({L,1...n}) %q) |− (forall ({M,L,-%p}) %q[%t/%p]);; (Var {L}) (Sent %q) (Term %t)

is a proof specification for any SCL universal quantification.

A more concrete example is the checking that the following rule is a correct application of `ndUI`:

> (forall (%a %b %c) %q) |− (forall (%a %c) %q["foo"/%b])

is a valid application of `ndUI` since (forall (%a %b %c) %q) "fits into"

> (forall ({L,1...n,%p}) %q) with $n = 3$ and $L = \{x_1 = \%a, x_2 = \%b, x_3 = \%c\}$

and (forall (%a %c) %q["box"/%p]) "fits into"

> (forall ({M,L,-%p}) %q[%t/%p]) with $M = L - \{x_2\} = \{x_1 = \%a, x_2 = \%b, x_3 = \%c\} - \{x_2\} = \{x_1 = \%a, x_3 = \%c\}$.

2.4 Syntax Conditions

Grammatical Categories

The specification of the grammatical category for a SCL schema placeholder is given by the following predicates. **we need to say something about these types**

- Sent(ence)
- Lit(eral)
- Clause
- Var(iable)
- Term

Unification

The following proof

GMP: {L,1...n}, (implies (and {L}) %q) |− %q ;; (Sent {L}, %q)

is an SCL proof schema for generalized modus ponens.

we GMP spec above is not complete yet since we still need to specify its side-condition unification

Clausification

The following proof

we need to add a PPDR proof schema for resolution

is an SCL proof schema for bi-resolution.

A sintax-condition should specify that the premises are clausified

2.5 Sentence Discharge

The following proof

ndImplIntro: %p |− (implies %q %p) ;; (Sent %p %q) ;; %q

shows that %q was discharged in order to introduce the implication in the proof schema conclusion. In a typical natural deduction proof, %q would be expected to be an assumption of %q. The sentence discharge approach here, although different, has the same effect since it discharges the occurrences of the sentence %p in all the premise assumptions.

For natural deduction or-elimination we have the following:

ndOrElim: (or %p %q) , %r, %r |− %r ;; (Sent %p %q %r) ;; %p, %q

In this case, the sentence %p was expected to be an assumption for one of the %r premises while the sentence %q was expected to be an assumption of the other %r premise. In PPDR, however, we do not need to specify where the assumptions are specified. So, the effect would be the same if %p and %q are assumptions on different %r premises and even if they are not assumptions for any premise at all. In fact, the PPDR discharge approach means that sentences are discharged if they are premise assumptions.

3 Proof Markup Language use of PPDR

I will describe the following in this section:

- **Inference Web and PML support for the PPDR specification of inference rules;**
- **An example of how PPDR specifications can be used for checking combined proofs (proofs produced by different agents?).**

4 Related Work

5 Conclusions

So, what is needed in the metalanguage is:

1. ways of indicating schemas by using metavariables over syntactic SCL classes. We can do this using KIF notation itself with a predicate for each syntax category, as illustrated.

2. a way to say that an assumption is 'closed' by a rule, and I think this need get no more complicated than the ORElim case above (that's the worst Ive ever seen)

3. a way to talk about substitutions of expressions for expressions within other expressions.

References

[1] Michael R. Genesereth and Richard Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, 1992.

[2] Deborah L. McGuinness and Paulo Pinheiro da Silva. Registry-Based Support for Information Integration. In *Proceedings of IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 117–122, Acapulco, Mexico, August 2003.

[3] Paulo Pinheiro da Silva, Deborah L. McGuinness, and Richard Fikes. A Proof Markup Language for Semantic Web Services. Technical Report KSL-04-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, January 2004.