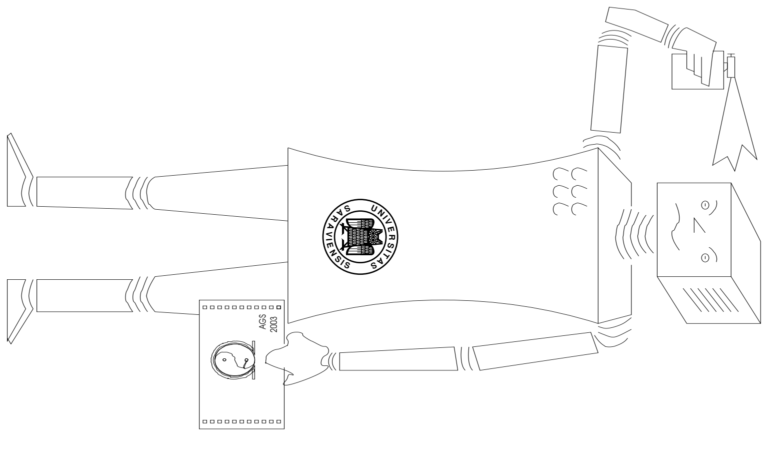
SEKI-REPORT ISSN 1437-4447

UNIVERSITÄT DES SAARLANDES
FACHRICHTUNG INFORMATIK
D–66123 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

# A Proof Representation

Jessi Berkelhammer
FR Informatik, Saarland Univ.
jessi@cmu.edu

SEKI Report SR–2004–02



**This SEKI Report was internally reviewed by:**
Christoph Benzmüller
FR Informatik, Universität des Saarlandes, D–66123 Saarbrücken, Germany
E-mail: chris@ags.uni-sb.de
WWW: www.ags.uni-sb.de/~chris/

**Editor of SEKI series:**
Claus-Peter Wirth
FR Informatik, Universität des Saarlandes, D–66123 Saarbrücken, Germany
E-mail: cp@ags.uni-sb.de
WWW: http://www.ags.uni-sb.de/~cp/welcome.html

# A Proof Representation

Jessi Berkelhammer
FR Informatik, Saarland Univ.
jessi@cmu.edu

August 23, 2004

**Abstract**

We present the proof representation language to be used by the mediator between an environment for writing mathematical documents and proof assistants. This language is an updated version of the one introduced in [1].

## 1 Motivation

We present an updated version of the proof representation language introduced in [1]. This language was introduced for the DIALOG project [2] as a user-oriented proof representation language which allows for under-specification. This updated version allows one to represent a greater variety of proofs; for example, proofs in which the user introduces an intermediary goal and decomposes that, and then proceeds with the proof. The representation language has been implemented in LISP.

The motivation was to give a formal representation of informal proofs. The representation aims to capture how proofs are constructed by humans. One key aspect of informal proof construction is under-specification; many references in a proof will not be specified by the user, and it is important to have a representation structure which allows for proofs in which some information is missing. The proof steps of the data structure correspond to the linguistically characterizable proofs steps done by humans. For example, one might work forward and give a new fact which has been derived from existing facts, or decompose the goal into subgoals, or introduce an assumption in order to start a subproof which will allow one to assert some additional formula. A proof is thus a set of linearized proof steps—one step follows from one or more other steps until the goal has been asserted. These proof steps may correspond to branches or backward work in a proof; the steps are linearized although some steps may point to subproofs or goal decompositions. There is a tension between informal and formal representations; we intend to give a representation which is formal enough to bridge the gap to the formal representations used by arbitrary proof assistants, but general enough to account for the expected linguistic assertions, which will be less precise than those generated by automated processes.

This updated proof language is presented in Fig. 1.

The updated proof checking rules are given in Fig. 2.

$$
\begin{array}{llcl}
\text{Step} & S & ::= & . \\
 & & & \mid \texttt{Trivial} \\
 & & & \mid \texttt{Fact}\ N : F\ \texttt{from}\ R^{*}; S \\
 & & & \mid \texttt{Subgoals}\ (N : F)^{+}\ \texttt{in}\ S^{+}\ \texttt{to obtain}\ N' : F'\ \texttt{by}\ R^{*}; S' \\
 & & & \mid \texttt{Assume}\ H^{*}\ \texttt{prove}\ N : F\ \texttt{in}\ S\ \texttt{to obtain}\ N' : F'\ \texttt{by}\ R^{*}; S' \\
 & & & \mid \texttt{Assign}\ (\mathit{SUBST} \mid \mathit{ABBRV}); S \\
 & & & \mid \texttt{Or}(S_1 \parallel \ldots \parallel S_n); S' \\
 & & & \mid \texttt{Cases}\ F^{+} : (\texttt{Case}\ N : F : S\ \texttt{End})^{+}\ \texttt{to obtain}\ N' : F'; S' \\
\text{Hypotheses} & H & ::= & N : F \mid \mathit{CONST} : \mathit{TYPE}? \mid \mathit{VAR} : \mathit{TYPE}? \\
\text{Substitutions} & \mathit{SUBST} & ::= & \texttt{Let}\ \mathit{VAR} := \mathit{TERM} \\
\text{Abbreviations} & \mathit{ABBRV} & ::= & \texttt{Let}\ \mathit{CONST} := \mathit{TERM} \\
\text{Constants} & \mathit{CONST} & ::= & \text{const}\ N \\
\text{Variables} & \mathit{VAR} & ::= & \text{var}\ N \\
\text{Types} & \mathit{TYPE} & ::= & \ldots
\end{array}
$$

Figure 1: Proof representation language

$$
\frac{P : \Gamma \Longrightarrow_{\mathsf{Triv}} \Delta}{\Gamma\langle \texttt{Trivial} \rangle \Delta}\ \mathsf{Trivial}
\qquad
\frac{\Gamma\langle S_i, S' \rangle \Delta}{\Gamma\langle \texttt{Or}(S_1 \parallel \ldots \parallel S_n); S' \rangle \Delta}\ \mathsf{Or}
$$

$$
\frac{P(R^{*}) : \Gamma \Longrightarrow_{\mathsf{Fact}} F, \Delta \quad \Gamma, N : F\langle S \rangle \Delta}{\Gamma\langle \texttt{Fact}\ N : F\ \texttt{from}\ R^{*}; S \rangle \Delta}\ \mathsf{Fact}
$$

$$
\frac{P(R^{*}) : \Gamma, (F_1 \wedge \ldots \wedge F_k) \Longrightarrow_{\mathsf{Subgoal}} F', \Delta \quad \Gamma\langle S_1 \rangle N_1 : F_1, \Delta \quad \ldots \quad \Gamma\langle S_k \rangle N_k : F_k, \Delta \quad \Gamma, N' : F'\langle S' \rangle \Delta}{\Gamma\langle \texttt{Subgoals}\ N_1 : F_1, \ldots, N_k : F_k\ \texttt{in}\ S_1 \mid \ldots \mid S_k\ \texttt{to obtain}\ N' : F'\ \texttt{by}\ R^{*}; S' \rangle \Delta}\ \mathsf{Subgoals}
$$

$$
\frac{P(R^{*}) : \Gamma, F \Longrightarrow_{\mathsf{Focus}} F', \Delta \quad P'(R^{*}) : \Gamma \Longrightarrow_{\mathsf{Hyp}} H_1 \wedge \ldots \wedge H_n, F', \Delta \quad \Gamma, H_1, \ldots, H_n\langle S \rangle N : F, \Delta \quad \Gamma, N' : F'\langle S' \rangle \Delta}{\Gamma\langle \texttt{Assume}\ H_1, \ldots, H_n\ \texttt{prove}\ N : F\ \texttt{in}\ S\ \texttt{to obtain}\ N' : F'\ \texttt{by}\ R^{*}; S' \rangle \Delta}\ \mathsf{Assume}
$$

$$
\frac{\texttt{var}\ x : \tau \in \Gamma \quad \Gamma \Longrightarrow_{\mathsf{Type}} t : \tau \quad P : \Gamma \Longrightarrow_{\mathsf{Subst}} x = t, \Delta \quad \Gamma, . : x = t\langle S \rangle \Delta}{\Gamma\langle \texttt{Assign var}\ x := t; S \rangle \Delta}\ \mathsf{Assign\text{-}Subst}
$$

$$
\frac{\Gamma \Longrightarrow_{\mathsf{Type}} t : \tau \quad c \notin \Gamma \cup \Delta \quad \Gamma, . : \texttt{const}\ c : \tau, . : c = t\langle S \rangle \Delta}{\Gamma\langle \texttt{Assign const}\ c := t; S \rangle \Delta}\ \mathsf{Assign\text{-}Abbrv}
$$

$$
\frac{P : \Gamma \Longrightarrow_{\mathsf{Case}} F_1 \vee \ldots \vee F_n, F', \Delta \quad \Gamma, N_1 : F_1\langle S_1 \rangle N' : F', \Delta \quad \ldots \quad \Gamma, N_n : F_n\langle S_n \rangle N' : F', \Delta \quad \Gamma, N' : F'\langle S' \rangle \Delta}{\Gamma\langle \texttt{Cases}\ F_1, \ldots, F_n : \texttt{Case}\ N_1 : F_1 : S_1\ \texttt{End} \ldots \texttt{Case}\ N_n : F_n : S_n\ \texttt{End to obtain}\ N' : F'; S' \rangle \Delta}\ \mathsf{Case}
$$

Figure 2: Proof Checking Rules

## 2 Explanation

One asserts the step `Trivial` when the current proof or subproof is complete. A proof, or subproof, is not complete unless the last step is a trivial step, indicating that the proven or obtained fact is the goal of the open proof or subproof. The step `Fact` $N : F$ `from` $R^*; S$ indicates that fact $F$, named $N$, has been derived using facts in $R^*$, and this step is followed by step $S$. Any of these fields may be under-specified. A proof step `Subgoals` $(N : F)^+$ `in` $S^+$ `to obtain` $N' : F'$ `by` $R^*; S'$ indicates that we have introduced a list of subgoals $(N : F)^+$, each of which has a proof in the corresponding list of subproofs $S^+$, and the facts given in $R^*$ show that this allows us to obtain fact $F'$, named $N'$. Fact $F'$ is a goal which we have decomposed into subgoals, but is not necessarily our original goal. $S'$ indicates the following step in the proof, which will be a trivial step in the case where $F'$ is the original goal, and otherwise this step will continue the proof of the original goal, with fact $F'$ available as a valid fact. A proof step `Assume` $H^*$ `prove` $N : F$ `in` $S$ `to obtain` $N' : F'$ `by` $R^*; S'$ introduces hypotheses $H^*$ in order to prove fact $F$ (named $N$) in subproof $S$. By reference to facts in $R^*$, this allows us to obtain fact $F'$ in the original proof, and continue the proof with step $S'$. For example, we might introduce some hypothesis $F_1$ in order to prove a contradiction within the subproof, and by referencing the appropriate rules of logic in $R^*$, this will allow to obtain the negation of $F_1$ at the level of the assumption step. A proof step `Assign` $(SUBST \mid ABBRV); S$ substitutes a variable for a term or abbreviates a term as a constant, and is followed by proof step $S$. The step `Or`$(S_1 \parallel \ldots \parallel S_n); S'$ indicates that there are different possible proofs, or subproofs, for some part of a proof, and these branches are followed by the subsequent step $S'$, which will be a trivial step in the situation where the different branches are possible proofs of the original goal. Case distinctions are introduced with a case step, `Cases` $F^+$ : (`Case` $N : F : S$ `End`)$^+$ `to obtain` $N' : F'; S'$, where for each case $F^x$ there is a subproof in $S^x$ which allows us to obtain the formula $F'$. The case split is followed by step $S'$.

### 2.1 Proof Checking

This representation is designed to be general and abstract, and thus does not assume a particular logic which establishes validity. Thus, proof checking is parameterized over the choice of a logic. A proof is checked by checking the individual proof steps of which it is composed. The proof checking rules give the lemmas that arise when one asserts a proof step of each category, and these lemmas must then be checked in the specific logic of choice to verify the proof step.

In these rules, the visible hypotheses are denoted by $\Gamma$ and the previous goals by $\Delta$. These rules not only check that some proof of a step's validity exists, but that such a proof uses the given references. For example, checking the step `Fact` $N : F$ `from` $R^*; S$ generates a **Fact** lemma, which ensures the validity of deriving fact $F$ from proof $P$ using the references in $R^*$, and generates a call to check if the following step, $S$, is valid.

## 3 Example

In [1], a student worked through a proof with the goal:

$$K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$$

(where "K" stands for "complement".)

Here are the student's utterances:

**S1:** by deMorgan-Rule-2 $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ holds

**S2:** $K(A \cup B)$ is $K(A) \cap K(B)$ according to deMorgan-1

**S3:** and $K(C \cup D)$ is also $K(C) \cap K(D)$ according to deMorgan-1

**S4:** hence follows finally: $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$.

Here is a print-out of the representation of this proof:

```
GOAL (K ((A V B) & (C V D)) = (K (A) & K (B)) V (K (C) & K (D)))
| |
NAME S1
FACT (K ((A V B) & (C V D)) = K (A V B) V K (C V D))
FROM DEMORGAN2
| |
NAME S2
FACT (K (A V B) = K (A) & K (B))
FROM DEMORGAN1
| |
NAME S3
FACT (K (C V D) = K (C) & K (D))
FROM DEMORGAN1
| |
NAME S4
FACT (K ((A V B) & (C V D)) = (K (A) & K (B)) V (K (C) & K (D)))
FROM (SUB S1 S2 S3)
TRIVIAL LABELNIL
```

The first line corresponds to the goal of the proof, and the four fact steps correspond to the user's four utterances. In the first three steps, the user only referenced the rule name (the different deMorgan rules). In the fourth fact step, the user did a substitution using the facts from the three previous steps (S1, S2, and S3). However, there was under-specification in the utterance, for the user's only justification was "hence follows finally."

In this proof, the user worked forward asserting facts which were derived from previous facts. A fact step is represented with certain characteristics—a label, the derived fact and its name, references to used facts, and references to the subsequent step. The proof itself has a goal and a first step, which points to the subsequent step, and so on. This proof is completed when the next step property of a fact step is a trivial step, indicating that the proven fact is the goal.

## References

[1] Serge Autexier, Christophe Benzmüller, Armin Fielder, Helmut Horacek, and Bao Quoc Vo. Assertion-level proof representation with under-specification. *Electronic Notes in Theoretical Computer Science*, 93:5–23, 2003.

[2] Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayova, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. Tutorial dialogs on mathematical proofs. In *Proceedings of IJCAI-03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, pages 12–22, Acapulco, Mexico, 2003.