Preuves Interactives et Applications

Christine Paulin & Burkhart Wolff

http://www.lri.fr/~paulin/PreuvesInteractives

Université Paris-Saclay

HOL and its Specification Constructs
Revisions

• What is “typed $\lambda$-calculus”
• What is “$\beta$-reduction”
• Using typed $\lambda$-calculus to represent logical systems
• What is “natural deduction” ? (from another perspective)
Revisions: Typed $\lambda$-calculus

• Rules over a global signature $\Sigma$:

$$ (c, \tau_c) \in \Sigma \quad \frac{\rho \vdash C : \tau_c}{\rho, x : \tau \vdash t : \sigma} \quad \frac{\rho \vdash \lambda x. t : \tau \rightarrow \sigma}{\rho \vdash f : \tau \rightarrow \sigma \quad \rho \vdash t : \tau} \quad \frac{X_1 : \tau_1, \ldots, X_p : \tau_p \vdash X_i : \tau_i}{\rho \vdash f t : \sigma} $$

• We assume $\Sigma =$

$$(\_+_\_\_, \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}), (\_0\_, \text{nat}), (\_\text{Suc} \_\_, \text{nat} \rightarrow \text{nat}), (\_=_\_, \text{nat} \rightarrow \text{nat} \rightarrow \text{bool}), (\_\text{True}\_, \text{bool}), (\_\text{False}\_, \text{bool}), (\_=_\_, \text{bool} \rightarrow \text{bool} \rightarrow \text{bool})$$
Revisions: Typed $\lambda$-calculus

• Examples: Are there variable environments $\rho$ such that the following terms are typable in $\Sigma$: (note that we use infix notation: we write “$0 + x$” instead of “$+_\_ 0 x$“)

- $(_+\_ 0) = (\text{Suc} \ x)$
- $((x + y) = (y + x)) = \text{False}$
- $f(_+\_ 0) = (\lambda c. \ g c) \ x$
- $+_\_ z (+_\_ (\text{Suc} 0)) = (0 + f \text{ False})$
- $a + b = (\text{True} = c)$
Revisions: $\beta$-reduction

- Assume that we want to find typed solutions for $?X, ?Y, ?Z$ such that the following terms become equivalent modulo $\alpha$-conversion and $\beta$-reduction:
  - $?X \ a \ =?= \ a + ?Y$
  - $(\lambda c. \ g \ c) \ =?= \ (\lambda x. \ ?Y \ x)$
  - $(\lambda c. \ ?X \ c) \ a \ =?= \ ?Y$
  - $\lambda a. \ (\lambda c. \ X \ c) \ a \ =?= \ (\lambda x. \ ?Y)$

- Note: Variables like $?X, ?Y, ?Z$ are called schematic variables; they play a major role in Isabelles Rule-Instantiation Mechanism

- Are the solutions for schematic variables always unique?
Deduction

- Logic Whirl-Pool of the 20ies (Girard) as response to foundational problems in Mathematics

  - growing uneasiness over the question:

    What is a proof?

    Are there limits of provability?
Deduction

• Historical context in the 20ies:
  – 1500 false proofs of „all parallels do not intersect in infinity“
  – lots of proofs and refutations of „all polyhedrons are eularian“ (Lakatosz)
  – Frege’s axiomatic set theory proven inconsistent by Russel
  – Science vs. Marxism debate (Popper)

\[ E = F + K - 2 \]
Deduction

• Historical context in the 20ies:
  – this seemed quite far away from Leipnitz vision of
    „Calculemus!“ (We don`t agree? Let`s calculate ...)
    of what constitutes, well,
    Science ...
Deduction

• Historical context in the 20ies:
  – attempts to formalize the intuition of „deduction“ by Frege, Hilbert, Russel, Lukasiewics, ...
  – 2 Calculi presented by Gerhard Gentzen in 1934.

• „natürliches Schliessen“ (natural deduction):

• „Sequenzkalkül“ (sequent calculus)

\[
\begin{align*}
\Gamma \vdash A \lor B & \quad \Gamma \cup \{A\} \vdash C & \quad \Gamma \cup \{B\} \vdash C \\
\hline
\Gamma \vdash C
\end{align*}
\]
Deduction

- An Inference System (or Logical Calculus) allows to infer formulas from a set of elementary judgements (axioms) and inferred judgements by rules:

\[
\begin{array}{c}
A_1 \quad \ldots \quad A_n \\
\hline
A_{n+1}
\end{array}
\]

“from the assumptions \( A_1 \) to \( A_n \), you can infer the conclusion \( A_{n+1} \).” A rule with \( n=0 \) is an elementary fact. Variables occurring in the formulas \( A_n \) can be arbitrarily substituted.
Deduction

- judgements discussed in this course (or elsewhere):

  \( t : \tau \) \quad "term \( t \) has type \( \tau \)"
  \( \Gamma \vdash \phi \) \quad "formula \( \phi \) is valid under assumptions \( \Gamma \)"
  \( \vdash \{ P \} \ x:= x+1 \ \{ Q \} \) \quad "Hoare Triple"

  \( \phi \) prop \quad "\( \phi \) is a property"
  \( \phi \) valid \quad "\( \phi \) is a valid (true) property"
  \( x \) mortal \( \implies \) sokrates mortal \quad --- judgements with free variable

  etc ...
Natural Deduction

- An Inference System for the equality operator (or “HO Equational Logic”) looks like this:

\[
\text{(where the first rule is an elementary fact).}
\]
Natural Deduction

- the same thing presented a bit more neatly (without prop):

\[ \frac{x = x}{s = t} \quad \frac{t = s}{r = s \quad s = t} \quad \frac{r = s \quad s = t}{r = t} \]

\[ \frac{\forall x. s \ x = t \ x}{s = t} \quad \frac{s = t \quad P s}{P t} \]

(equality on functions as above ("extensional equality") is an HO principle, and it is a classical principle).
Representing logical systems in the typed $\lambda$-calculus

• It is straightforward to use the typed $\lambda$-terms as a syntactic means to represent logics; including binding issues related to quantifiers like $\forall$, $\exists$, ...

• Example: The Isabelle language „Pure“:
It consists of typed $\lambda$-terms with constants:
  – foundational types “prop” and “_ => _” (“_ $\Rightarrow$ _”)
  – the Pure (universal) quantifier
    all :: “(\alpha \rightarrow Prop) \rightarrow Prop”
    (“$\forall x. P x$”, “\<And> x. P x” “!!x. P x”)
  – the Pure implication “A ===> B” (“_ $\Longrightarrow$ _”)
  – the Pure equality “A == B” “A $\equiv$ B”
Pure: A (Meta)-Language for Deductive Systems

- Pure is a language to write logical rules.
- Wr. Isabelle, it is the meta-language, i.e. the built-in formula language.
- Equivalent notations for natural deduction rules:

$$\begin{align*}
  A_1 \implies (\ldots \implies (A_n \implies A_{n+1})\ldots),
  \hline
  \[A_1; \ldots; A_n\] \implies A_{n+1},
  \hline
  \begin{array}{c}
  A_1 \\
  \ldots \\
  A_n \\
  \hline
  \end{array}
  \Rightarrow A_{n+1}
\end{align*}$$

Theorem

assumes $A_1$

and $A_n$

shows $A_{n+1}$
„Pure“: A (Meta)-Language for Deductive Systems

• Some more complex rules involving the concept of “Discharge” of (formerly hypothetical) assumptions:

\[(P \rightarrow Q) \rightarrow R : \]

theorem
assumes "P \rightarrow Q"
shows "R"

\[
\begin{array}{c}
[P] \\
\vdash \\
Q \\
\hline
R
\end{array}
\]
Propositional Logic as ND calculus

• Some (almost) basic rules in HOL

\[
\begin{align*}
\frac{Q}{\neg \neg \neg \neg \neg Q} & \quad \frac{\neg \neg Q}{Q} \quad \frac{\neg \neg \neg \neg \neg B}{\neg \neg \neg \neg \neg A \rightarrow B} \\
\text{notnotE} & \quad \text{impI} & \quad \text{mp}
\end{align*}
\]

\[
\begin{align*}
\frac{A}{A \lor B} & \quad \frac{A \lor B}{Q} \\
\text{disjI1} & \quad \text{disjI2}
\end{align*}
\]
Propositional Logic as ND calculus

- Some (almost) basic rules in HOL

\[
\begin{align*}
\frac{A \land B}{Q} & \quad \text{conjE} \\
\frac{Q}{A \land B} & \quad \text{conjI}
\end{align*}
\]

\[
\begin{align*}
[A, B] \\
\cdot \\
\cdot \\
\end{align*}
\]
Key Concepts: Rule-Instances

- A Rule-Instance is a rule where the free variables in its judgements were substituted by a common substitution $\sigma$:

\[
\begin{align*}
\frac{A \quad B}{A \land B} & \text{conjI} \\
\sigma & \\
\frac{3 < x \quad x \leq y}{3 < x \land x \leq y}
\end{align*}
\]

where $\sigma$ is $\{A \mapsto 3 < x, B \mapsto x \leq y\}$. 
Key Concepts: Formal Proofs

- A series of inference rule instances is usually displayed as a Proof Tree (or: Derivation or: Formal Proof)

\[
\begin{align*}
\text{sym} & \quad f(a, b) = a & \quad f(a, b) = a & \quad f(f(a, b), b) = c \\
\hline
a = f(a, b) & \quad f(a, b) = c & \quad \text{trans} & \quad g(a) = g(a) \\
\hline
\text{refl} & \quad a = c & \quad \text{subst} & \quad g(a) = g(c)
\end{align*}
\]

- The hypothetical facts at the leaves are called the assumptions of the proof (here \( f(a, b) = a \) and \( f(f(a, b), b) = c \)).
Key Concepts: Discharge

- A key requisite of ND is the concept of **discharge** of assumptions allowed by some rules (like \( \text{impI} \)).

\[
\begin{align*}
&\frac{[f(a, b) = a]}{\text{sym}} \quad \frac{[f(a, b) = a]}{\text{sym}} \quad f(f(a, b), b) = c \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \ quad
Key Concepts: Global Assumptions

- The set of (proof-global) assumptions gives rise to the notation:

\[ \{ f(a, b) = a, f(f(a, b), b) = c \} \vdash g(a) = g(c) \]

written:

\[ A \vdash \phi \]

or when emphasising the global theory (also called: global context):

\[ A \vdash_E \phi \]
Sequent-style calculus

- Gentzen introduced and alternative “style” to natural deduction: Sequent style rules.
  - Idea: using the tuples $A \vdash \phi$ as basic judgments of the rules.

\[
\begin{align*}
\Gamma, A & \vdash B \\
\hline
\Gamma & \vdash A \rightarrow B
\end{align*}
\]
Sequent-style calculus

- in contrast to:

\[
\begin{align*}
\frac{[A]}{B} & \quad \frac{A \rightarrow B}{B} \\
\vdots & \\
\end{align*}
\]
Sequent-style vs. ND calculus

- Both styles are linked by two transformations called “lifting over assumptions.” Lifting over assumptions transforms:

\[
\begin{array}{c}
A_1 \ldots A_n \\
\hline
A_{n+1}
\end{array}
\]

where we consider for the moment \( \vdash \) just equivalent to meta implication \( \rightarrow \).
Quantifiers

- When reasoning over logics with quantifiers (such as FOL, set-theory, TLA, ..., and of course: HOL), the additional concept of "parameters" of a rule is necessary. We assume that there is an infinite set of variables and that it is always possible to find a "fresh" unused one ...

- Consider:

\[ \forall x. P(x) \]

for any term \( t \)

\[ P(t) \]

\[ \forall x. P(x) \]

for any fresh variable \( u \)

\[ P(u) \]

\[ [P(y)]_y \]

\[ \cdots \]

\[ \forall x. P(x) \]

\[ Q \]

\[ Q \]

\[ [P(n)]_n \]

\[ \cdots \]

\[ P(0) \]

\[ P(Suc \ n) \]

\[ \forall x. P(x) \]
Quantifiers

- For all I, Isabelle allows certain free variables ?X, ?Y, ?Z that represent "wholes" in a term that can be filled in later by substitution; Coq requires the instantiation when applying the rule.

- Isabelle uses a built-in ("meta")-quantifier $\forall x. P x$ already seen on page 13; Coq uses internally a similar concept not explicitly revealed to the user.
Introduction to Isabelle/HOL
Basic HOL Syntax

• HOL (= Higher-Order Logic) goes back to Alonzo Church who invented this in the 30ies ...

• “Classical” Logic over the $\lambda$-calculus with Curry-style typing (in contrast to Coq)

• Logical type: “bool” injects to “prop”. i.e

  \[ \text{Trueprop} :: \text{“bool $\Rightarrow$ prop”} \]

is wrapped around any HOL-Term without being printed:

  \[ \text{Trueprop A $\Rightarrow$ Trueprop B} \] is printed: \( A \Rightarrow B \) but \( A::\text{bool} \)!
Basic HOL Syntax

• Logical connective syntax (Unicode + ASCII):
  input: print: alt-ascii input

  – “_ \<and>_ _” “_\^_” “_& _”
  – “_ \<or>_ _” “_\_v_” “_ |_ ”
  – “_ \<longrightarrow>_ _” “_ \rightarrow_ _” “_ --> _”
  – “_ \<not>_ _” “_\neg_” “_~ _”
  – “\<forall> x. P“ “\forall x. P” “! x. P x”
  – “\<exists> x. P“ “\exists x. P” “? x. P x”
Basic HOL Rules

• HOL is an equational logic, i.e. a system with the constant “_=_::'a 'a bool” and the rules:

\[
\begin{align*}
\text{refl} &: x = x \\
\text{sym} &: \frac{s = t}{t = s} \\
\text{trans} &: \frac{r = s \quad s = t}{r = t}
\end{align*}
\]

\[
\begin{align*}
\text{ext} &: \frac{\forall x. s \ x = t \ x}{s = t} \\
\text{subst} &: \frac{s = t \quad P s}{P t}
\end{align*}
\]
Basic HOL Rules

- HOL is an equational logic, i.e. a system with the constant "\_\_::'a 'a bool" and the rules:

\[
\begin{align*}
\text{refl} & : x = x \\
\text{sym} & : s = t \Rightarrow t = s \\
\text{trans} & : r = s \Rightarrow s = t \Rightarrow r = t
\end{align*}
\]

\[
\begin{align*}
\text{ext} & : \forall x. s \ x = t \ x \\
\text{subst} & : s = t \quad P \ s \Rightarrow P \ t
\end{align*}
\]

which rule makes HOL "higher-order"??
Basic HOL Rules

• Some (almost) basic rules in HOL

\[
\begin{align*}
\text{conjI} & : \\
A \land B & \quad Q \\ 
\hline \\
& Q \\
\text{conjE} & : \\
A \land B & \quad A \quad B \\
\hline \\
& A \land B
\end{align*}
\]
The quantifier rules of HOL.

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]

- The quantifier rules of HOL.
  \[ \forall x. P \quad \Rightarrow \quad \forall x. P \]
  \[ \exists x. Q \quad \Rightarrow \quad \exists x. Q \]
HOL Rules

- The quantifier rules of HOL:

\[
\begin{array}{c}
[P \ ?t; \forall x. P \ x] \\
\vdots \\
\forall x. P \ x \\
\hline
Q
\end{array}
\]

\[Q\]

\text{alldupE (unsafe, but complete)}
HOL Rules

• The quantifier rules of HOL:

\[
\begin{array}{c}
\forall x. P \ x \\
\vdots \\
\forall x. P \ x \\
\hline
Q
\end{array}
\]

\[
\begin{array}{c}
P \ ?t; \forall x. P \ x \\
\vdots \\
Q
\end{array}
\]

alldupE (unsafe, but complete)
HOL Rules

• The quantifier rules of HOL:

\[
\begin{align*}
P \ ?t & \quad \text{exl} \\
\exists x. P & \quad x \\
\end{align*}
\]

\[
\begin{align*}
[ P(x) ]_x & \\
\exists x. P(x) & \quad Q \\
\end{align*}
\]

exE
HOL Rules

• From these rules (which were defined actually slightly differently), a large body of other rules can be DERIVED (formally proven, and introduced as new rule in the proof environment).

Examples: see exercises.
Typed Set-theory in HOL

• The HOL Logic comes immediately with a typed set - theory: The type

\[ \alpha \text{ set} \equiv \alpha \Rightarrow \text{bool}, \text{ that's it!} \]

can be defined isomorphically to its type of characteristic functions!

• THIS GIVES RISE TO A RICH SET THEORY DEVELOPPED IN THE LIBRARY (Set.thy).
Typed Set Theory: Syntax

• Logical connective syntax (Unicode + ASCII):

input:
“\in”
“\{.\}”
“\cup”
“\cap”
“\subseteq”

print:
“∈”
“{.}”
“∪”
“∩”
“⊆”

alt-ascii input
“:\:”
“\{.\}”
“\cup”
“\cap”
“\subseteq”

for example
“\{x. True \land x = x\}”

“\cup”
“\cap”
“\subseteq”
Conclusion

• Typed $\lambda$-calculus is a rich term language for the representation of logics, logical rules, and logical derivations (proofs)

• On the basis of typed $\lambda$-calculus, Higher-order logic (HOL) is fairly easy to represent

• ... the differences to first-order logic (FOL) are actually tiny.