A humble presentation of ELAN

Robin Andersson
Ingemar Axelsson
Joakim Mjärdner
Main features

• Semantics based on many-sorted algebras

• Possibility to express strategies of rewrite rule application

• User-definable syntax

• Non-deterministic computations and user defined strategy language to control rewriting

• Pre-processor/Interpreter/Compiler

• REF exchange format
Three levels of programming

Speciﬁcation

User

Logic

Super user

Rewriting

Theory

pre-processing

Query

Interprete

RESULTS

End user

Query

Interpreter

RESULTS

User
Rules in ELAN

The rewrite rules may be labelled and may have local variables

\[ [lab] : l(x) \Rightarrow r(y) \text{ if } v \text{ where } y := [S]u(x) \]

where \( lab \) is the rule label, \( v \) the condition and \( y := [S]u(x) \) the local affectation assigned, to the local variable \( y \), one of the results of the strategy \( S \) applied to the term \( u \).

[expand] \((p_1+p_2)*p_3 \Rightarrow e_1+e_2\)

where \( e_2:=(\text{simplify}) \ p_2*p_3 \)
where \( e_1:=(\text{simplify}) \ p_1*p_3 \) end
An example: quicksort

module quick

import global int list[int];
end
sort pair;
end
operators global
    sort(@) : (list[int]) list[int];
    pivot(@,@,@,@) : (int list[int] list[int] list[int]) pair;
    [@,@] : (list[int] list[int]) pair;
end

rules for list[int]
x : int;
    xs,s,l : list[int];
global
    [] sort(nil) => nil
end
\[
\begin{align*}
\text{sort}(x.xs) & \Rightarrow \text{append}(\text{sort}(s),x.\text{sort}(l)) \\
& \quad \text{where } (\text{pair})[s,l] := ()\text{pivot}(x,xs,nil,nil)
\end{align*}
\]

end

end

rules for pair
p, x : int;
xs, s, l : list[int];
R : pair;
global
\[
\begin{align*}
\text{pivot}(p,\text{nil},s,l) & \Rightarrow [s,l] \\
\end{align*}
\]

end

\[
\begin{align*}
\text{pivot}(p,x.xs,s,l) & \Rightarrow R \\
& \quad \text{choose} \\
& \quad \text{try if } p < x \text{ where } R := () \text{pivot}(p,xs,s,x.l) \\
& \quad \text{try where } R := () \text{pivot}(p,xs,x.s,l)
\end{align*}
\]

end

end

end
Computation and deduction in ELAN

leftmost-innermost  (Call-by value) normalisation (default)

Computations  are described with unlabelled rules

Deductions  are described by labelled rules

Strategies  for deductions are themselves described using labelled or unlabelled rewrite rules
Strategies

Sequential composition

\[ S_1; S_2 \]

Iteration

repeat*(S) or iterate*(S)

Choice points

\[ \text{dk}(S_1 \ldots S_n) : \text{all results of all } S_i \text{ (Don't know choose)} \]
\( dc(S_1 \ldots S_n) \): all results of one \( S_i \) (Don’t care choose)

\( \text{first}(S_1 \ldots S_n) \): all results of the first \( S_i \)

**Cut points**

\( dc \text{ one}(S_1 \ldots S_n) \): one results of one \( S_i \)

\( \text{first} \text{ one}(S_1 \ldots S_n) \): one results of the first \( S_i \)
An example strategy

[factorize]  \[ P + (p1*p2) + (p1*p3) \rightarrow P + p4 \]
  where \( p4 := (\text{simplify}) \ p1*(p2+p3) \)  
end

[expand]  \[ P + (p1+p2)*p3 \rightarrow P + e1+e2 \]
  where \( e2 := (\text{simplify}) \ p2*p3 \)
  where \( e1 := (\text{simplify}) \ p1*p3 \)  
end

::

[.]  \text{simplify} \rightarrow
  \text{repeat}*(\text{first one}(\text{expand})) ;
  \text{repeat}*(\text{first one}(\text{factorise}))
end
Applications

- Theorem provers (B-tools)
- Completion procedures for rewriting systems (Knuth Bendix)
- Automata computations
- CP on finite domains (COLETTE)
- Unifications