Implementing Deep Inference

24.11.04, LORIA, Nancy

Ozan Kahramanoğulları

Institut für Informatik, Universität Leipzig
International Center for Computational Logic, TU Dresden
ozan@informatik.uni-leipzig.de
Outline

• Motivation
• The Calculus of Structures & System BV
• From Derivations to Rewritings
• The Implementation
• Tuning the Performance: System BVn & System BVu
• Performance Comparison
• Conclusion and Future Work
Motivation:

– general recipe for implementing deductive systems with deep inference
– tools for developing the proof theory of different logics
– implementing logic BV: a self-dual, non-commutative operator

• Process Algebra:
  
  \( a, b \) processes: \( a.b \neq b.a \)

  CCS vs. BV [Bruscoli, 02]

• Planning:
  
  \( \langle openDoor ; enterRoom \rangle \neq \langle enterRoom ; openDoor \rangle \)

  Conjunctive Planning Problems [K, AIA’05]

• (Natural Language Processing)
The Calculus of Structures & System BV

- [Guglielmi, 99] proof theoretical formalism, generalization of the one-sided sequent calculus
- **Deep inference:** Rules can be applied deep inside a structure.

\[
\frac{S\{R\}}{S\{T\}}
\]

- **System BV:**
  - \(\text{MLL} + \text{mix} + \text{mix}0 + \) a non-commutative, self-dual operator

- **BV structures:**
  - \(S ::= \circ \mid a \mid [S, \ldots, S] \mid (S, \ldots, S) \mid \langle S; \ldots; S \rangle \mid S^\dagger\)

- Structures are considered equivalent modulo an equational theory.
Syntactic Equivalence of BV Structures

**Associativity**

\[
[\bar{R}, [\bar{T}]] = [\bar{R}, T]
\]
\[
(R, (\bar{T})) = (\bar{R}, \bar{T})
\]
\[
\langle \langle \bar{R} \rangle ; \langle \bar{T} \rangle \rangle = \langle \bar{R}; \bar{T} \rangle
\]

**Commutativity**

\[
[\bar{R}, T] = [\bar{T}, \bar{R}]
\]
\[
(R, \bar{T}) = (\bar{T}, R)
\]

**Unit**

\[
(o, R) = R
\]
\[
\langle o ; R \rangle = R
\]

**Negation**

\[
o = o
\]
\[
[R, T] = (\bar{R}, \bar{T})
\]
\[
(R, T) = [\bar{R}, \bar{T}]
\]
\[
\langle R ; T \rangle = \langle \bar{R}; \bar{T} \rangle
\]

**Context Closure**

\[
\text{if } R = T \text{ then } S\{R\} = S\{T\}
\]
\[
\bar{R} = \bar{T}
\]

**System BV**

\[
\text{System BV}
\]

\[
\frac{S\{o\}}{S[a, \bar{a}]}
\]

\[
\frac{S([R, U], T)}{S[(R, T), U]}
\]

\[
\frac{S\langle [R, U]; [T, V] \rangle}{S[\langle R; T \rangle, \langle U; V \rangle]}
\]
From Derivations to Rewritings

(1.) Explicit equality steps

Example:

\[
\frac{[c, [\bar{b}, b]), \bar{c}] \quad \sim \quad [b, (\bar{b}, c), \bar{c}]}{s \quad = \quad [(c, [\bar{b}, b]), \bar{c}]} \quad = \quad [([\bar{b}, b], c), \bar{c}]} \quad = \quad [b, (\bar{b}, c), \bar{c}]
\]

(2.) From n-ary operators to binary operators

Example: \( n22([\bar{a}, b, (a, c)]) = [\bar{a}, [b, (a, c)]]) \)

(3.) From structures to terms

\[ \Sigma = \{\circ, \bar{\_}, <\_;\_>, [\_;\_], (\_;\_)) \cup \{a \mid \text{a is a positive atom}\} \]
From Derivations to Rewritings

From inference rules to rewrite rules

\[
\begin{align*}
\text{Associativity} & \quad \text{Commutativity} & \quad \text{Unit} \\
\langle R; \langle S; T \rangle \rangle & \approx \langle \langle R; S \rangle; T \rangle, & [R, T] & \approx [T, R], & \langle \circ; R \rangle & \approx R, \\
[R, [S, T]] & \approx [[R, S], T], & (R, T) & \approx (T, R), & \langle \circ, R \rangle & \approx R, \\
(R, (S, T)) & \approx ((R, S), T), & (R, T) & \approx (T, R), & (\circ, R) & \approx R.
\end{align*}
\]

\[
\begin{align*}
\text{R} = \begin{cases}
\text{ai} \downarrow: [a, \bar{a}] & \rightarrow \circ, \\
\text{s} : [(R, T), U] & \rightarrow ([R, U], T), \\
\text{q} \downarrow: [\langle R; R' \rangle, \langle T; T' \rangle] & \rightarrow \langle [R, T]; [R', T'] \rangle
\end{cases}
\end{align*}
\]
Orienting the Equalities for Negation

**Definition:** A $\Sigma$-term $s$ is in **negation normal form** iff the negation is pushed to the leaves (atoms) and no unit $\circ$ appears in it.

$$\mathcal{R}_{Neg} = \left\{ \begin{array}{ll} \overline{\circ} & \rightarrow \circ, \\ \overline{\langle R; T \rangle} & \rightarrow \langle \overline{R}; \overline{T} \rangle, \\ \overline{[R, T]} & \rightarrow (\overline{R}, \overline{T}), \\ \overline{(R, T)} & \rightarrow [\overline{R}, \overline{T}], \\ \overline{\overline{R}} & \rightarrow R \end{array} \right\}$$

**Lemma:** Term rewriting systems $\mathcal{R}_{Neg}$ is terminating and confluent.

**Lemma:** For $\Sigma$-term $s$, the normal form of $s$ with respect to $\mathcal{R}_{Neg}$ is in **negation normal form**.
From Derivations to Rewritings

\[ s \rightarrow_{R/E} t \quad \text{iff} \quad (\exists s', t') \quad s \approx_E s' \rightarrow_R t' \approx_E t \]

Example:

\[
\begin{align*}
S & : [b, (c, [a, \bar{a}])] \\
& \rightarrow [\bar{a}, b, (a, c)] \\
& \approx_E [\bar{a}, [b, (a, c)]]
\end{align*}
\]

\[
\begin{align*}
& \approx_E [b, ((a, c), \bar{a})] \rightarrow_R [b, ([a, \bar{a}], c)] \\
& \approx_E [b, (c, [a, \bar{a}])]
\end{align*}
\]

Proposition: Let \( s \) and \( t \) be two \( \Sigma \)-terms or structures, where \( t \) is in negation normal form.
There is a derivation in \( \text{BV} \) from \( s \) to \( t \) having length \( n \) iff there exists a rewriting \( s \stackrel{\Sigma}{\rightarrow R_{Neg}} s' \rightarrow_{R/E} t \).
Implementation in Maude

- Systems in the calculus of structures can be expressed as term rewriting systems modulo equational theories. [K, Hölldobler, TR-04]

- Language Maude allows implementing term rewriting systems modulo associativity, commutativity and unit(s). [K, ESSLLI’04]

- Since Maude 2 breadth-first search is available.
Maude Module for System BV

mod BV is
sorts Atom Unit Structure .
subsort Atom < Structure .
subsort Unit < Structure .

op o : -> Unit .
op _- : Atom -> Atom [ prec 50 ] .
ops a b c d e : -> Atom .

var R T U V : Structure . var A : Atom .

rl [q-down] : [ < R ; T > , < U ; V > ] => < [R,U] ; [T,V] > .
endm
Implementation in Maude

Maude> search [- c, [< a ; {c, - b} >, < - a ; b >]] =>+ o .

search in BV : [- c, [< a ; {c, - b} >, < - a ; b >]] =>+ o .
Solution 1 (state 2229)
states: 2230  rewrites: 196866 in 930ms cpu (950ms real) (211683 rewrites/second)
empty substitution

No more solutions.
states: 2438  rewrites: 306179 in 1460ms cpu (1490ms real) (209711 rewrites/second)

- Units cause redundant applications of the inference rules which do not lead to a proof.
Removing the Units: Some Definitions

A structure is in \textit{unit normal form} when the only negated structures appearing in it are atoms and no unit \( \circ \) appears in it.

Example: \( \langle [b, (\overline{a}, c), \circ, \overline{c}]; a \rangle = \langle ([a, \overline{c}], \overline{b}, c); \overline{a} \rangle \)

Two systems \( \mathcal{S} \) and \( \mathcal{S}' \) are \textit{equivalent} if for every proof of a structure \( T \) in system \( \mathcal{S} \), there exists a proof of \( T \) in system \( \mathcal{S}' \), and vice versa.
System BV  $\leadsto$  System BVn

$$[\circ, R] = R$$
$$\langle \circ ; R \rangle = R$$
$$\langle R ; \circ \rangle = R$$

$$\begin{align*}
\text{u}_1 \downarrow & \frac{S\{R\}}{S[R, \circ]} \\
\text{u}_2 \downarrow & \frac{S\{R\}}{S(R, \circ)} \\
\text{u}_3 \downarrow & \frac{S\{R\}}{S\langle R ; \circ \rangle} \\
\text{u}_4 \downarrow & \frac{S\{R\}}{S\langle \circ ; R \rangle}
\end{align*}$$

**Proposition:** Every BV structure $S$ can be transformed to one of its unit normal forms $S'$ by applying only the rules $\{\text{u}_1 \downarrow, \text{u}_2 \downarrow, \text{u}_3 \downarrow, \text{u}_4 \downarrow\}$ bottom-up and the equalities for negation from left to right.
System BV  \xrightarrow{\sim}  System BVn

\[
\begin{align*}
S([R, T], U) & \xrightarrow{S} S((R, U), T) \\
S\langle[R, T]; [U, V]\rangle & \xrightarrow{q} S\langle[R; U], \langle T; V]\rangle
\end{align*}
\]

Proposition: The rules \(q_1\downarrow, q_2\downarrow, q_3\downarrow, q_4\downarrow, s_1,\) and \(s_2\) are sound (derivable) for system BV.
Theorem: For every derivation $\Delta \vdash_{BV} W$ there exists a derivation $\Delta' \vdash_{BVn} W'$ where $W'$ is a unit normal form of the structure $W$.

Corollary: System $BV$ and system $BVn$ are equivalent.
Corollary: System BV and system BVu are equivalent.
Performance Comparison

Example (proof search): \([- c, [< a ; \{c, - b\} >, < - a ; b >]]\)

<table>
<thead>
<tr>
<th></th>
<th>finds a proof</th>
<th>search terminates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>in # millisec.</td>
<td>after # rewrites</td>
</tr>
<tr>
<td>BV</td>
<td>950</td>
<td>196866</td>
</tr>
<tr>
<td>BVn</td>
<td>120</td>
<td>12610</td>
</tr>
<tr>
<td>BVu</td>
<td>10</td>
<td>1416</td>
</tr>
</tbody>
</table>

Proposition: If a structure $R$ in unit normal form with $n$ occurrences of positive atoms has a proof in $BV_n$ with length $k$, then $R$ has a proof in $BV_u$ with length $k - n$. 
Conclusion

- System BV (and other systems in the calculus of structures) can be expressed as term rewriting systems modulo equational theories which can be implemented in Maude.
- **BVn, BVu**: systems equivalent to BV where equalities for unit become redundant. [K, ISCIS’04]
- These systems provide a **better performance** in proof search.
- These results can be analogously applied to **other systems** in the calculus of structures.
- Implementations are available for download at http://www.informatik.uni-leipzig.de/~ozan/maude_cos.html.
Future Work

- Using the expressive power of **TOM & Java** for implementing efficient search strategies.
- Applying **local search techniques** for more efficient proof search.
- Developing **proof theoretical strategies**, e.g., splitting theorem, together with other methods attached to rewriting calculus.