"If you have a bottom clause, you have almost solved the problem."

A Bottom Set Strategy for Tractable Feature Construction

Filip Železný

ČVUT **Prague**, School of Electrical Engineering, Dept. of Cybernetics
The Gerstner Laboratory for Intelligent Decision Making and Control

Introduction

:: We are concerned with constructing **features**, ie. expressions such as

`hasCar(C), hasLoad(C,L), isBig(L)`

:: The **language of features** $\mathcal{F}(n)$ is a set where elements comply to constraints:

– **Size**. Each $f \in \mathcal{F}(n)$ has at most n atoms.

– **Typing & Moding**. Specify predicates, arguments types and i/o modes.

```
hasCar(-car), hasLoad(+car,-load),
isBig(+load), isSmall(+load)
```

– **ETA Constraint**. Each var has both an input and an output occurrence. (ETA = 'Extended Transformation Approach' [Lavrač & Flach, 2001])

The Bottom Set Theorem

:: 2 Problems of size n

– **Existence**: Find **an** element of $\mathcal{F}(n)$.

– **Enumeration**: Find **all** elements of $\mathcal{F}(n)$ (up to var renaming).

:: A **bottom set** is an expression $\perp(n)$ complying to the typing constraint such that $f \subseteq \perp(n)$ (up to var renaming) whenever $f \in \mathcal{F}(n)$.

:: **Theorem**. Given a $\perp(n)$ such that $|\perp(n)| \leq poly(n)$, one can solve efficiently the existence problem, ie. find in time $poly(n)$ a feature $f \in \mathcal{F}(n)$ if it exists, or answer NO.

:: **Proof**. Naive approach: must check ETA constraint for $O(|\perp(n)|^n)$ subsets :-(.
Trick: reduce polynomially onto **HORNSAT**.

The Essence of the Proof in an Example

:: Bottom set

$$\perp(3) = \quad \texttt{hasCar}(\texttt{C}),\ \texttt{hasLoad}(\texttt{C},\texttt{L}),\ \texttt{isBig}(\texttt{L}),\ \texttt{isSmall}(\texttt{L})$$

propositional vars: P_1 P_2 P_3 P_4

:: HORNSAT assignment: P_i **false** iff i-th atom **included** in feature

variable	production	consumption
C	$P_2 \leftarrow P_1$	$P_1 \leftarrow P_2$
L	$P_3 \leftarrow P_2$	$P_2 \leftarrow P_3 \wedge P_4$
	$P_4 \leftarrow P_2$	
non-emptiness	$\leftarrow P_1 \wedge P_2 \wedge P_3 \wedge P_4$	

:: $P_1 = P_2 = P_3 = false$ is a **maximal** (fewest false assignments) **solution** (found efficiently [Dowling & Gallier, 1984]), thus $f = \texttt{hasCar}(\texttt{C}), \texttt{hasLoad}(\texttt{C},\texttt{L}), \texttt{isBig}(\texttt{L})$ is **a minimal feature**. Because $|f| \leq 3$, we output f (otherwise would say NO).

Tractable Cases: Example

:: A connected **depth-limited** language $\mathcal{F}_D(n) \subset \mathcal{F}(n)$

$$
\text{branching factor} \leq n \left\{
\begin{array}{llll}
\mathtt{hasCar}(\mathtt{C}) & \rightarrow \mathtt{hasRoof}(\mathtt{C}) & & \\
 & \rightarrow \mathtt{hasLoad}(\mathtt{C}, \mathtt{L1}) & \rightarrow \mathtt{isBig}(\mathtt{L1}) & \\
 & & \rightarrow \mathtt{isSmall}(\mathtt{L2}) & \\
 & \rightarrow \mathtt{hasLoad}(\mathtt{C}, \mathtt{L2}) & \rightarrow \mathtt{isBig}(\mathtt{L2}) & \\
 & & \rightarrow \mathtt{isSmall}(\mathtt{L2}) &
\end{array}
\right.
$$

$$\leftarrow \text{depth} \leq D \rightarrow$$

:: If literals with multiple inputs, then branching factor $O(n^A)$ ($A \sim$ max arity).

:: Therefore $|\perp(n)| \leq O(n^{A \times D}) = poly(n)$.

:: Therefore tractable.

Intractable Cases: Example (W-I-P)

:: A "no variable reuse" language $\mathcal{F}_R(n) \subset \mathcal{F}(n)$. Represent typing/moding by a matrix $\mathbf{M}$. Find a feature $f \in \mathcal{F}_R(n)$ a represent it by a vector $\vec{f}$. For example

$$\mathbf{M} = \begin{array}{|ccccc|l} \texttt{hasCar} & \texttt{hasLoad} & \texttt{hasRoof} & \texttt{isBig} & \texttt{isSmall} & \\ \hline -1 & +1 & +1 & 0 & 0 & \texttt{car} \\ 0 & -1 & 0 & +1 & +1 & \texttt{load} \\ \hline \end{array}$$

$f =$`hasCar(C1), hasLoad(C1,L), isBig(L), hasCar(C2), hasRoof(C2)`

$$\vec{f} = \begin{array}{ccccc} [\,2, & 1, & 1, & 1, & 0\,]^{-1} \\ \texttt{hasCar} & \texttt{hasLoad} & \texttt{hasRoof} & \texttt{isBig} & \texttt{isSmall} \end{array}$$

$\vec{f}$ solves (due to no-var-reuse assumption) the matrix equation

$$\mathbf{M} \times \vec{f} = \vec{0},$$

Ie. solves an instance of the NP-complete integer programming problem. Remains to show reduction from an arbitrary integer programming instance.

"If you have a bottom clause, you have almost solved the problem."

*** THE END ***