Script **generated by TTT**

Title: Seidl: Programmoptimierung (30.01.2013)

Date: Wed Jan 30 08:30:41 CET 2013

Duration: 89:52 min

Pages: 38

$$
\begin{aligned}
[\![\mathbf{match}\ e_0\ \mathbf{with}\ [\,] \to e_1 \mid x :: xs \to e_2]\!]^\sharp\,\rho &= \\
&\quad [\![e_0]\!]^\sharp\,\rho \wedge ([\![e_1]\!]^\sharp\,\rho \vee [\![e_2]\!]^\sharp\,(\rho \oplus \{x, xs \mapsto 1\})) \\
[\![\mathbf{match}\ e_0\ \mathbf{with}\ (x_1, x_2) \to e_1]\!]^\sharp\,\rho &= \\
&\quad [\![e_0]\!]^\sharp\,\rho \wedge [\![e_1]\!]^\sharp\,(\rho \oplus \{x_1, x_2 \mapsto 1\}) \\
[\![[\,]]\!]^\sharp\,\rho = [\![e_1 :: e_2]\!]^\sharp\,\rho = [\![(e_1, e_2)]\!]^\sharp\,\rho &= 1
\end{aligned}
$$

- The rules for **match** are analogous to those for **if**.
- In case of ::, we know nothing about the values beneath the constructor; therefore $\{x, xs \mapsto 1\}$.
- We check our analysis on the function app ...

Example:

$$
\begin{aligned}
\mathsf{app} = \ & \mathbf{fun}\ x \to \mathbf{fun}\ y \to \mathbf{match}\ x\ \mathbf{with}\ [\,] \to y \\
& \mid x :: xs \to x :: \mathsf{app}\ xs\ y
\end{aligned}
$$

Abstract interpretation yields the system of equations:

$$
\begin{aligned}
[\![\mathsf{app}]\!]^\sharp\, b_1\, b_2 &= b_1 \wedge (b_2 \vee 1) \\
&= b_1
\end{aligned}
$$

We conclude that we may conclude for sure only for the first argument that its top constructor is required :-)

Example:

$$
\begin{aligned}
\mathsf{app} = \ & \mathbf{fun}\ x \to \mathbf{fun}\ y \to \mathbf{match}\ x\ \mathbf{with}\ [\,] \to y \\
& \mid x :: xs \to x :: \mathsf{app}\ xs\ y
\end{aligned}
$$

Abstract interpretation yields the system of equations:

$$
\begin{aligned}
[\![\mathsf{app}]\!]^\sharp\, b_1\, b_2 &= b_1 \wedge (b_2 \vee 1) \\
&= b_1
\end{aligned}
$$

We conclude that we may conclude for sure only for the first argument that its top constructor is required :-)

860

app x y

Example:

$$\textsf{app} \;=\; \textbf{fun}\ x \to \textbf{fun}\ y \to \textbf{match}\ x\ \textbf{with}\ [\,] \to y$$
$$\mid\ x :: xs \to x :: \textsf{app}\ xs\ y$$

Abstract interpretation yields the system of equations:

$$[\![\textsf{app}]\!]^\sharp\, b_1\, b_2 \;=\; b_1 \wedge \cancel{(b_2 \vee 1)}\ 1$$
$$= b_1$$

We conclude that we may conclude for sure only for the first argument that its top constructor is required :-)

Example:

$$\textsf{app} \;=\; \textbf{fun}\ x \to \textbf{fun}\ y \to \textbf{match}\ x\ \textbf{with}\ [\,] \to y$$
$$\mid\ x :: xs \to x :: \textsf{app}\ xs\ y$$

Abstract interpretation yields the system of equations:

$$[\![\textsf{app}]\!]^\sharp\, b_1\, b_2 \;=\; b_1 \wedge (b_2 \vee 1)$$
$$= b_1$$

We conclude that we may conclude for sure only for the first argument that its top constructor is required :-)

860

Total Strictness

Assume that the result of the function application is totally required. Which arguments then are also totally required ?

We again refer to Boolean functions ...

$$[\![\textbf{match}\ e_0\ \textbf{with}\ [\,] \to e_1 \mid x, :: xs \to e_2]\!]^\sharp\,\rho \;=\; \textbf{let}\ b = [\![e_0]\!]^\sharp\,\rho\ \textbf{in}$$
$$b \wedge [\![e_1]\!]^\sharp\,\rho \vee [\![e_2]\!]^\sharp\,(\rho \oplus \{x \mapsto b, xs \mapsto 1\}) \vee [\![e_2]\!]^\sharp\,(\rho \oplus \{x \mapsto 1, xs \mapsto b\})$$
$$[\![\textbf{match}\ e_0\ \textbf{with}\ (x_1, x_2) \to e_1]\!]^\sharp\,\rho \;=\; \textbf{let}\ b = [\![e_0]\!]^\sharp\,\rho\ \textbf{in}$$
$$[\![e_1]\!]^\sharp\,(\rho \oplus \{x_1 \mapsto 1, x_2 \mapsto b\}) \vee [\![e_1]\!]^\sharp\,(\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 1\})$$
$$[\![[\,]]\!]^\sharp\,\rho \;=\; 1$$
$$[\![e_1 :: e_2]\!]^\sharp\,\rho \;=\; [\![e_1]\!]^\sharp\,\rho \wedge [\![e_2]\!]^\sharp\,\rho$$
$$[\![(e_1, e_2)]\!]^\sharp\,\rho \;=\; [\![e_1]\!]^\sharp\,\rho \wedge [\![e_2]\!]^\sharp\,\rho$$

let rec f x = f (x+1)

$$[f]^\# b = [f]^\# (b \wedge 1)$$

	$[f]^\#$	
b	fun b	→ 0

Total Strictness

Assume that the result of the function application is totally required.
Which arguments then are also totally required ?

We again refer to Boolean functions ...

$$
\begin{array}{lcl}
\llbracket \mathbf{match}\ e_0\ \mathbf{with}\ [\,]\ \to\ e_1\ \mid\ x::xs\ \to\ e_2 \rrbracket^\sharp\ \rho & = & \mathbf{let}\ b = \llbracket e_0 \rrbracket^\sharp\ \rho\ \mathbf{in} \\
\multicolumn{3}{l}{b \wedge \llbracket e_1 \rrbracket^\sharp\ \rho \vee \llbracket e_2 \rrbracket^\sharp\ (\rho \oplus \{x \mapsto b, xs \mapsto 1\}) \vee \llbracket e_2 \rrbracket^\sharp\ (\rho \oplus \{x \mapsto 1, xs \mapsto b\})} \\
\llbracket \mathbf{match}\ e_0\ \mathbf{with}\ (x_1, x_2)\ \to\ e_1 \rrbracket^\sharp\ \rho & = & \mathbf{let}\ b = \llbracket e_0 \rrbracket^\sharp\ \rho\ \mathbf{in} \\
\multicolumn{3}{r}{\llbracket e_1 \rrbracket^\sharp\ (\rho \oplus \{x_1 \mapsto 1, x_2 \mapsto b\}) \vee \llbracket e_1 \rrbracket^\sharp\ (\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 1\})} \\
\llbracket [\,] \rrbracket^\sharp\ \rho & = & 1 \\
\llbracket e_1 :: e_2 \rrbracket^\sharp\ \rho & = & \llbracket e_1 \rrbracket^\sharp\ \rho \wedge \llbracket e_2 \rrbracket^\sharp\ \rho \\
\llbracket (e_1, e_2) \rrbracket^\sharp\ \rho & = & \llbracket e_1 \rrbracket^\sharp\ \rho \wedge \llbracket e_2 \rrbracket^\sharp\ \rho
\end{array}
$$

Total Strictness

Assume that the result of the function application is totally required.
Which arguments then are also totally required ?

We again refer to Boolean functions ...

$$
\begin{array}{lcl}
\llbracket \mathbf{match}\ e_0\ \mathbf{with}\ [\,]\ \to\ e_1\ \mid\ x::xs\ \to\ e_2 \rrbracket^\sharp\ \rho & = & \mathbf{let}\ b = \llbracket e_0 \rrbracket^\sharp\ \rho\ \mathbf{in} \\
\multicolumn{3}{l}{b \wedge \llbracket e_1 \rrbracket^\sharp\ \rho \vee \llbracket e_2 \rrbracket^\sharp\ (\rho \oplus \{x \mapsto b, xs \mapsto 1\}) \vee \llbracket e_2 \rrbracket^\sharp\ (\rho \oplus \{x \mapsto 1, xs \mapsto b\})} \\
\llbracket \mathbf{match}\ e_0\ \mathbf{with}\ (x_1, x_2)\ \to\ e_1 \rrbracket^\sharp\ \rho & = & \mathbf{let}\ b = \llbracket e_0 \rrbracket^\sharp\ \rho\ \mathbf{in} \\
\multicolumn{3}{r}{\llbracket e_1 \rrbracket^\sharp\ (\rho \oplus \{x_1 \mapsto 1, x_2 \mapsto b\}) \vee \llbracket e_1 \rrbracket^\sharp\ (\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 1\})} \\
\llbracket [\,] \rrbracket^\sharp\ \rho & = & 1 \\
\llbracket e_1 :: e_2 \rrbracket^\sharp\ \rho & = & \llbracket e_1 \rrbracket^\sharp\ \rho \wedge \llbracket e_2 \rrbracket^\sharp\ \rho \\
\llbracket (e_1, e_2) \rrbracket^\sharp\ \rho & = & \llbracket e_1 \rrbracket^\sharp\ \rho \wedge \llbracket e_2 \rrbracket^\sharp\ \rho
\end{array}
$$

861

Discussion:

- The rules for constructor applications have changed.
- Also the treatment of **match** now involves the components z and x_1, x_2.
- Again, we check the approach for the function app.

Example:

Abstract interpretation yields the system of equations:

$$
\begin{array}{lcl}
\llbracket \mathsf{app} \rrbracket^\sharp\ b_1\ b_2 & = & b_1 \wedge b_2 \vee b_1 \wedge \llbracket \mathsf{app} \rrbracket^\sharp\ 1\ b_2 \vee 1 \wedge \llbracket \mathsf{app} \rrbracket^\sharp\ b_1\ b_2 \\
& = & b_1 \wedge b_2 \vee b_1 \wedge \llbracket \mathsf{app} \rrbracket^\sharp\ 1\ b_2 \vee \llbracket \mathsf{app} \rrbracket^\sharp\ b_1\ b_2
\end{array}
$$

Example:

$$
\begin{array}{lcl}
\mathsf{app} & = & \mathbf{fun}\ x\ \to\ \mathbf{fun}\ y\ \to\ \mathbf{match}\ x\ \mathbf{with}\ [\,]\ \to\ y \\
& & \quad \mid\ x::xs\ \to\ x::\mathsf{app}\ xs\ y
\end{array}
$$

Abstract interpretation yields the system of equations:

$$
\begin{array}{lcl}
\llbracket \mathsf{app} \rrbracket^\sharp\ b_1\ b_2 & = & b_1 \wedge (b_2 \vee 1) \\
& = & b_1
\end{array}
$$

We conclude that we may conclude for sure only for the first argument that its top constructor is required :-)

Discussion:

- The rules for constructor applications have changed.
- Also the treatment of **match** now involves the components z and x_1, x_2.
- Again, we check the approach for the function app.

Example:

Abstract interpretation yields the system of equations:

$$\begin{array}{lll} [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee \cancel{1 \wedge}\ [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\ & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \end{array}$$

Discussion:

- The rules for constructor applications have changed.
- Also the treatment of **match** now involves the components z and x_1, x_2.
- Again, we check the approach for the function app.

Example:

Abstract interpretation yields the system of equations:

$$\begin{array}{lll} [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee 1 \wedge [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\ & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\ & = & b_1 \wedge b_2 \vee b_1 \wedge 0 \vee 0 \end{array}$$

This results in the following fixpoint iteration:

0	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to 0$
1	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to x \wedge y$
2	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to x \wedge y$

We deduce that both arguments are definitely totally required if the result is totally required :-)

Warning:

Whether or not the result is totally required, depends on the context of the function call!

In such a context, a specialized function may be called ...

Discussion:

- The rules for constructor applications have changed.
- Also the treatment of **match** now involves the components z and x_1, x_2.
- Again, we check the approach for the function app.

Example:

Abstract interpretation yields the system of equations:

$$\begin{array}{lll} [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee 1 \wedge [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\ & = & b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\ & = & b_1 b_2 \vee b_1 b_2 \vee b_1 \wedge b_2 \end{array}$$

This results in the following fixpoint iteration:

0	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to 0$
1	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to x \wedge y$
2	$\mathbf{fun}\ x \to \mathbf{fun}\ y \to x \wedge y$

We deduce that both arguments are definitely totally required if the result is totally required :-)

Warning:

Whether or not the result is totally required, depends on the context of the function call!

In such a context, a specialized function may be called ...

let rec f x = f(x+1)

$[f]^\# b = [f]^\# b$

	$[f]^\#$
0	fun $b \to 0$

let rec f i = if f(inc i+1)

$$\begin{array}{lll} \mathsf{app}\# = \mathbf{fun}\ x \to \mathbf{fun}\ y \to & \mathbf{let}\ \#x' = x\ \mathbf{and}\ \#y' = y\ \mathbf{in} & \\ & \mathbf{match}\ 'x\ \mathbf{with}\ [\,] \to\ 'y' & \\ & \mid\ x :: xs \to\ \ \mathbf{let}\ \#r = x :: \mathsf{app}\#\ xs\ y & \\ & \mathbf{in}\ r & \end{array}$$

total :

$[f]^\# b = b \wedge [f]^\# b$

Discussion

- Both strictness analyses employ the same complete lattice.
- Results and application, though, are quite different :-)
- ~~Thereby, we use the following description relations:~~

head

Top Strictness	:	$\bot \,\Delta\, 0$
Total Strictness	:	$z \,\Delta\, 0$ if $\bot$ occurs in z.

$[f]^\# b = 1$

- Both analyses can also be combined to an a joint analysis ...

Combined Strictness Analysis

- We use the complete lattice:

$$\mathbb{T} = \{0 \sqsubset 1 \sqsubset 2\}$$

- The description relation is given by:

$$\bot \,\Delta\, 0 \qquad z \,\Delta\, 1\ (z \text{ contains } \bot) \qquad z \,\Delta\, 2\ (z \text{ value})$$

- The lattice is more informative, the functions, though, are no longer as efficiently representable, e.g., through Boolean expressions :-(
- We require the auxiliary functions:

$$(i \sqsubseteq x);\ y = \begin{cases} y & \text{if } i \sqsubseteq x \\ 0 & \text{otherwise} \end{cases}$$

The Combined Evaluation Function:

$$
\begin{aligned}
[\![\mathbf{match}\, e_0\, \mathbf{with}\, [\,] \to e_1 \mid x :: xs \to e_2]\!]^\sharp \rho \;&=\; \mathbf{let}\; b = [\![e_0]\!]^\sharp \rho\; \mathbf{in} \\
&\qquad (2 \sqsubseteq b)\,;\, [\![e_1]\!]^\sharp \rho \sqcup \\
&\qquad (1 \sqsubseteq b)\,;\, ([\![e_2]\!]^\sharp (\rho \oplus \{x \mapsto 2, xs \mapsto b\}) \\
&\qquad\qquad \sqcup [\![e_2]\!]^\sharp (\rho \oplus \{x \mapsto b, xs \mapsto 2\})) \\
[\![\mathbf{match}\, e_0\, \mathbf{with}\, (x_1, x_2) \to e_1]\!]^\sharp \rho \;&=\; \mathbf{let}\; b = [\![e_0]\!]^\sharp \rho\; \mathbf{in} \\
&\qquad (1 \sqsubseteq b)\,;\, ([\![e_1]\!]^\sharp (\rho \oplus \{x_1 \mapsto 2, x_2 \mapsto b\}) \\
&\qquad\qquad \sqcup [\![e_1]\!]^\sharp (\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 2\})) \\
[\![\,[\,]\,]\!]^\sharp \rho \;&=\; 2 \\
[\![e_1 :: e_2]\!]^\sharp \rho \;&=\; \\
[\![(e_1, e_2)]\!]^\sharp \rho \;&=\; 1 \sqcup ([\![e_1]\!]^\sharp \rho \sqcap [\![e_2]\!]^\sharp \rho)
\end{aligned}
$$

Example:

For our beloved function app, we obtain:

$$
\begin{aligned}
[\![\mathsf{app}]\!]^\sharp\, d_1\, d_2 \;&=\; (2 \sqsubseteq d_1)\,;\, d_2 \sqcup \\
&\qquad (1 \sqsubseteq d_1)\,;\, (1 \sqcup [\![\mathsf{app}]\!]^\sharp\, d_1\, d_2 \sqcup d_1 \sqcap [\![\mathsf{app}]\!]^\sharp\, 2\, d_2) \\
&=\; (2 \sqsubseteq d_1)\,;\, d_2 \sqcup \\
&\qquad (1 \sqsubseteq d_1)\,;\, 1 \sqcup \\
&\qquad (1 \sqsubseteq d_1)\,;\, [\![\mathsf{app}]\!]^\sharp\, d_1\, d_2 \sqcup \\
&\qquad d_1 \sqcap [\![\mathsf{app}]\!]^\sharp\, 2\, d_2
\end{aligned}
$$

this results in the fixpoint computation:

0	$\mathbf{fun}\; x \to \mathbf{fun}\; y \to\; 0$
1	$\mathbf{fun}\; x \to \mathbf{fun}\; y \to\; (2 \sqsubseteq x);\; y \sqcup (1 \sqsubseteq x);\; 1$
2	$\mathbf{fun}\; x \to \mathbf{fun}\; y \to\; (2 \sqsubseteq x);\; y \sqcup (1 \sqsubseteq x);\; 1$

We conclude

- that both arguments are totally required if the result is totally required; and
- that the root of the first argument is required if the root of the result is required :-)

Remark:

The analysis can be easily generalized such that it guarantees evaluation up to a depth d ;-)

$f\,(g\,x) = g\,(g\,x)$

Further Directions:

- Our Approach is also applicable to other data structures.
- In principle, also higher-order (monomorphic) functions can be analyzed in this way :-)
- Then, however, we require higher-order abstract functions — of which there are many :-(
- Such functions therefore are approximated by:

$$\mathbf{fun}\; x_1 \;\to\; \ldots\; \mathbf{fun}\; x_r \;\to\; \top$$

2^2

:-)

- For some known higher-order functions such as map, foldl, loop, ... this approach then should be improved :-))

Further Directions:

- Our Approach is also applicable to other data structures.
- In principle, also higher-order (monomorphic) functions can be analyzed in this way :-)
- Then, however, we require higher-order abstract functions — of which there are many :-(
- Such functions therefore are approximated by:

$$\textbf{fun}\ x_1 \ \to\ \ldots\ \textbf{fun}\ x_r\ \to\ \top$$

:-)

- For some known higher-order functions such as map, foldl, loop, ... this approach then should be improved :-))

5 Optimization of Logic Programs

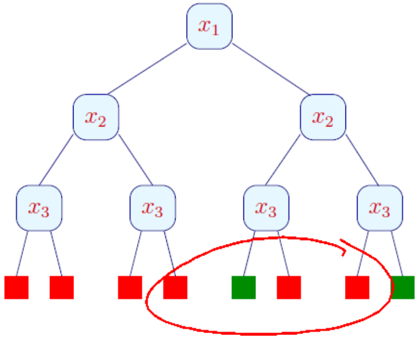
We only consider the mini language PuP ("Pure Prolog"). In particular, we do not consider:

- arithmetic;
- the cut-operator.
- Self-modification by means of assert and retract.

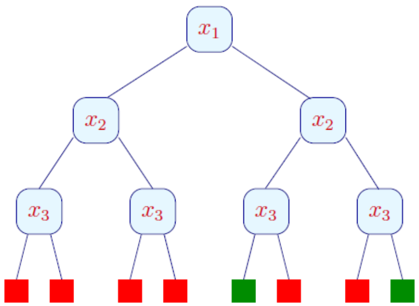
Example:

$$\begin{array}{lll}
\text{bigger}(X,Y) & \leftarrow & X = \textit{elephant}, Y = \textit{horse} \\
\text{bigger}(X,Y) & \leftarrow & X = \textit{horse}, Y = \textit{donkey} \\
\text{bigger}(X,Y) & \leftarrow & X = \textit{donkey}, Y = \textit{dog} \\
\text{bigger}(X,Y) & \leftarrow & X = \textit{donkey}, Y = \textit{monkey} \\
\text{is_bigger}(X,Y) & \leftarrow & \text{bigger}(X,Y) \\
\text{is_bigger}(X,Y) & \leftarrow & \text{bigger}(X,Z), \text{is_bigger}(Z,Y) \\
 & \leftarrow & \text{is_bigger}(\textit{elephant}, \textit{dog})
\end{array}$$

... yields the tree:

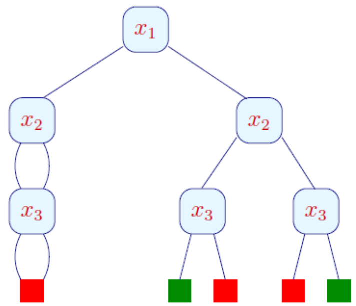


... yields the tree:

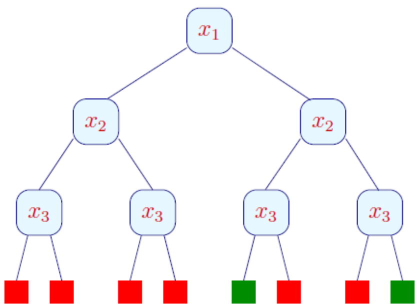


Idea (2):

- Decision trees are exponentially large :-(
- Often, however, many sub-trees are isomorphic :-)
- Isomorphic sub-trees need to be represented only once ...

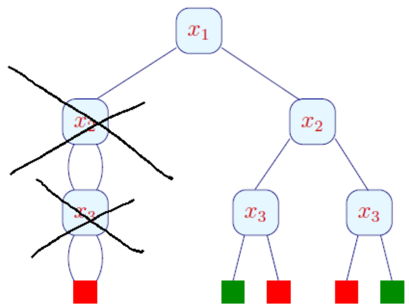


... yields the tree:



Idea (2):

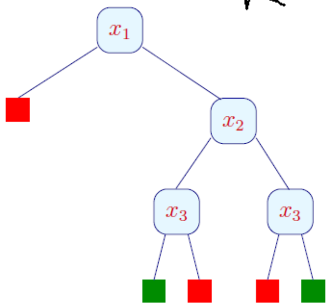
- Decision trees are exponentially large :-(
- Often, however, many sub-trees are isomorphic :-)
- Isomorphic sub-trees need to be represented only once ...



Idea (3):

- Nodes whose test is irrelevant, can also be abandoned ...

ROBDD



Discussion:

- This representation of the Boolean function f is unique !

  $\Longrightarrow$

  Equality of functions is efficiently decidable !!
- For the representation to be useful, it should support the basic operations: $\wedge, \vee, \neg, \Rightarrow, \exists\, x_j$ ...

$$\begin{array}{lll} [b_1 \wedge b_2]_k & = & b_1 \wedge b_2 \\ [f \wedge g]_{i-1} & = & \textbf{fun } x_i \to \textbf{if } x_i \textbf{ then } [f\,1 \wedge g\,1]_i \\ & & \qquad\quad \textbf{else } [f\,0 \wedge g\,0]_i \end{array}$$

// analogous for the remaining operators

Background 6: Binary Decision Diagrams

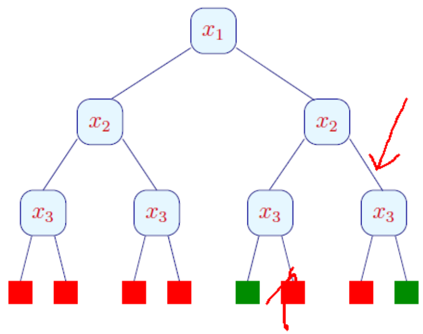
Idea (1):

- Choose an ordering $x_1, \ldots, x_k$ on the arguments ...
- Represent the function $f : \mathbb{B} \to \ldots \to \mathbb{B}$ by $[f]_0$ where:

$$\begin{array}{lll} [b]_k & = & b \\ [f]_{i-1} & = & \textbf{fun } x_i \to \textbf{if } x_i \textbf{ then } [f\ 1]_i \\ & & \qquad\quad \textbf{else } [f\ 0]_i \end{array}$$

Example: $f\ x_1\ x_2\ x_3 = x_1 \wedge (x_2 \leftrightarrow x_3)$

... yields the tree:



$$
\begin{aligned}
[\exists x_j.\, f]_{i-1} &= \textbf{fun } x_i \to \textbf{if } x_i \textbf{ then } [\exists x_j.\, f\ 1]_i \\
&\qquad\qquad \textbf{else } [\exists x_j.\, f\ 0]_i \qquad\qquad \text{if } i < j \\
[\exists x_j.\, f]_{j-1} &= [f\ 0 \vee f\ 1]_j
\end{aligned}
$$

- Operations are executed bottom-up.
- Root nodes of already constructed sub-graphs are stored in a unique-table

  $\Longrightarrow$

  Isomorphy can be tested in constant time !
- The operations thus are polynomial in the size of the input BDDs :-)

Discussion:

- Originally, BDDs have been developped for circuit verification.
- Today, they are also applied to the verification of software ...
- A system state is encoded by a sequence of bits.
- A BDD then describes the set of all reachable system states.
- Warning: Repeated application of Boolean operations may increase the size dramatically !
- The variable ordering may have a dramatic impact ...

Example: $(x_1 \leftrightarrow x_2) \wedge (x_3 \leftrightarrow x_4)$