Beispiel:

$$\begin{array}{lcl} \mathsf{app}(X, Y, Z) & \leftarrow & X = [\,],\; Y = Z \\ \mathsf{app}(X, Y, Z) & \leftarrow & X = [H|X'],\; Z = [H|Z'],\; \mathsf{app}(X', Y, Z') \end{array}$$

Dann ist:

$$\begin{array}{lcl} [\![\mathsf{app}]\!]^\sharp(X) & \sqsupseteq & X \wedge (Y \leftrightarrow Z) \\ [\![\mathsf{app}]\!]^\sharp(X) & \sqsupseteq & \textbf{let } \psi = X \wedge H \wedge X' \wedge (Z \leftrightarrow Z') \\ & & \textbf{in } \exists\, H, X', Z'.\; \mathsf{combine}^\sharp_{\dots}\,(\psi, [\![\mathsf{app}]\!]^\sharp(\mathsf{enter}^\sharp_{\dots}\,(\psi))) \end{array}$$

wobei für $\psi = X \wedge H \wedge X' \wedge (Z \leftrightarrow Z')$:

$$\begin{array}{lcl} \mathsf{enter}^\sharp_{\dots}\,(\psi) & = & X \\ \mathsf{combine}^\sharp_{\dots}\,(\psi, X \wedge (Y \leftrightarrow Z)) & = & (X \wedge H \wedge X' \wedge (Z \leftrightarrow Z') \wedge (Y \leftrightarrow Z') \end{array}$$

Beispiel (Forts.):

Weiterhin haben wir:

$$
\begin{array}{lll}
[\![\mathsf{app}]\!]^\sharp(Z) & \sqsupseteq & X \wedge Y \wedge Z \\
[\![\mathsf{app}]\!]^\sharp(Z) & \sqsupseteq & \mathbf{let}\ \psi = X \wedge H \wedge X' \wedge Z \wedge Z' \\
 & & \mathbf{in}\ \ \exists\, H, X', Z'.\ \mathsf{combine}^\sharp_{\dots}(\psi, [\![\mathsf{app}]\!]^\sharp(\mathsf{enter}^\sharp_{\dots}(\psi)))
\end{array}
$$

wobei für $\psi = Z \wedge H \wedge Z' \wedge (X \leftrightarrow X')$:

$$
\begin{array}{lll}
\mathsf{enter}^\sharp_{\dots}(\psi) & = & Z \\
\mathsf{combine}^\sharp_{\dots}(\psi, X \wedge Y \wedge Z) & = & X \wedge H \wedge X' \wedge Y \wedge Z \wedge Z'
\end{array}
$$

Die Fixpunkt-Iteration liefert damit:

$$
[\![\mathsf{app}]\!]^\sharp(X) \ = \ X \wedge (Y \leftrightarrow Z) \qquad [\![\mathsf{app}]\!]^\sharp(Z) \ = \ X \wedge Y \wedge Z
$$

Diskussion:

- Vollständige Tabellierung der Transformationen $[\![\mathsf{app}]\!]^\sharp$ ist nicht praktikabel.
- Wir setzen darum bedarfsgetriebene Fixpunktiteration ein !
- Die Auswertung startet mit der Auswertung der Anfrage g, d.h. mit der Auswertung von $[\![g]\!]^\sharp\,1$.
- Die Menge der betrachteten Fixpunkt-Variablen $[\![p]\!]^\sharp\,\psi$ liefert eine Beschreibung der möglichen Aufrufe :-))
- Zur effizienten Repräsentation von Funktionen $\psi \in \mathsf{Pos}$ eignen sich binäre Entscheidungsdiagramme (BDDs).

Exkurs 5: Binäre Entscheidungsdiagramme

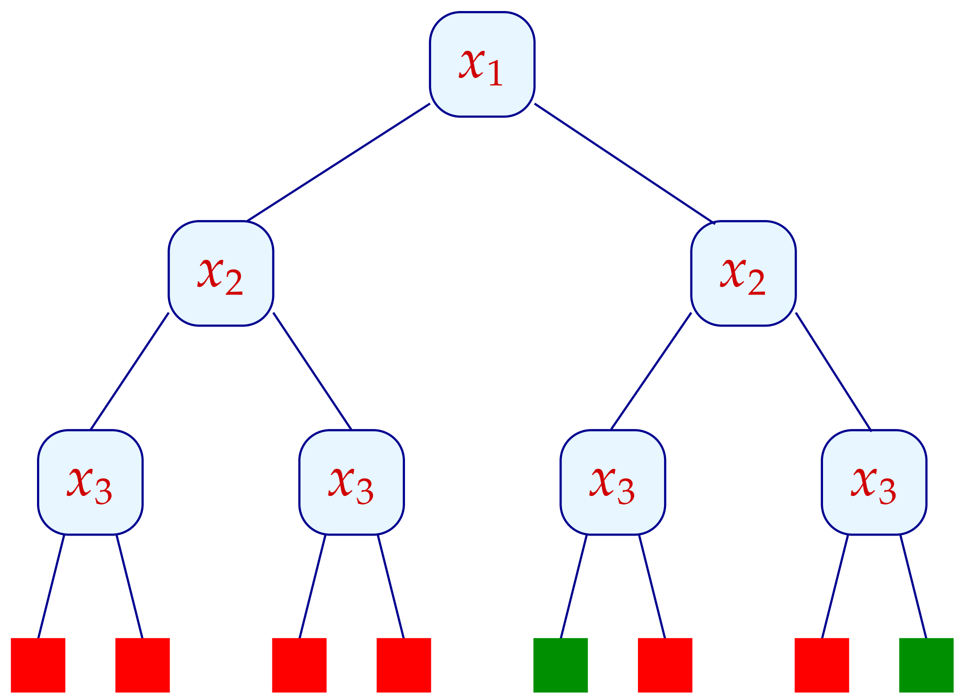
Idee (1):

- Wähle eine Anordnung $x_1, \ldots, x_k$ auf den Argumenten ...
- Repräsentiere die Funktion $f : \mathbb{B} \to \ldots \to \mathbb{B}$ durch $[f]_0$ wobei:

$$\begin{array}{lcl} [b]_k & = & b \\ [f]_{i-1} & = & \textbf{fn}\ x_i \Rightarrow \textbf{if}\ x_i\ \textbf{then}\ [f\ 1]_i \\ & & \qquad\qquad \textbf{else}\ [f\ 0]_i \end{array}$$

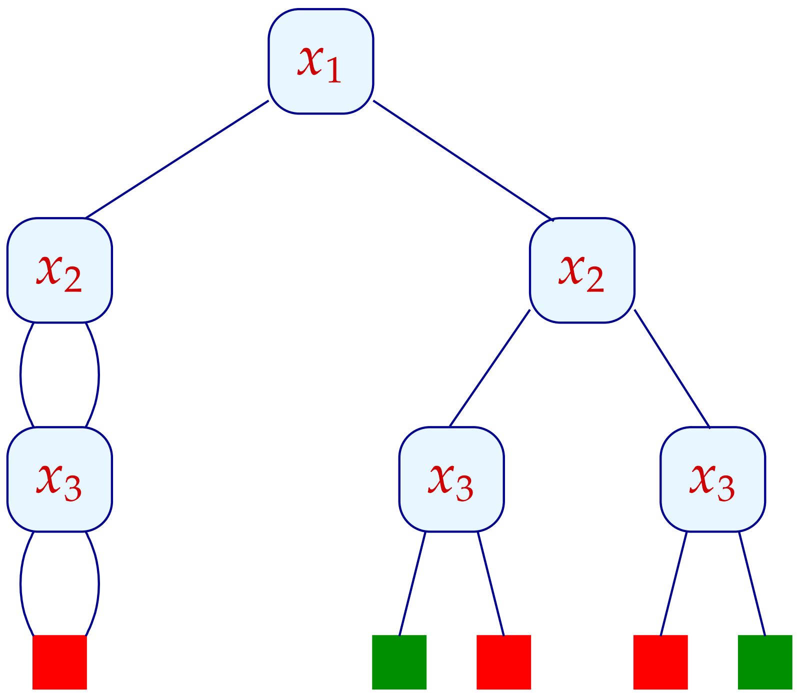
Beispiel: $f\ x_1\ x_2\ x_3 = x_1 \wedge (x_2 \leftrightarrow x_3)$

... liefert den Baum:



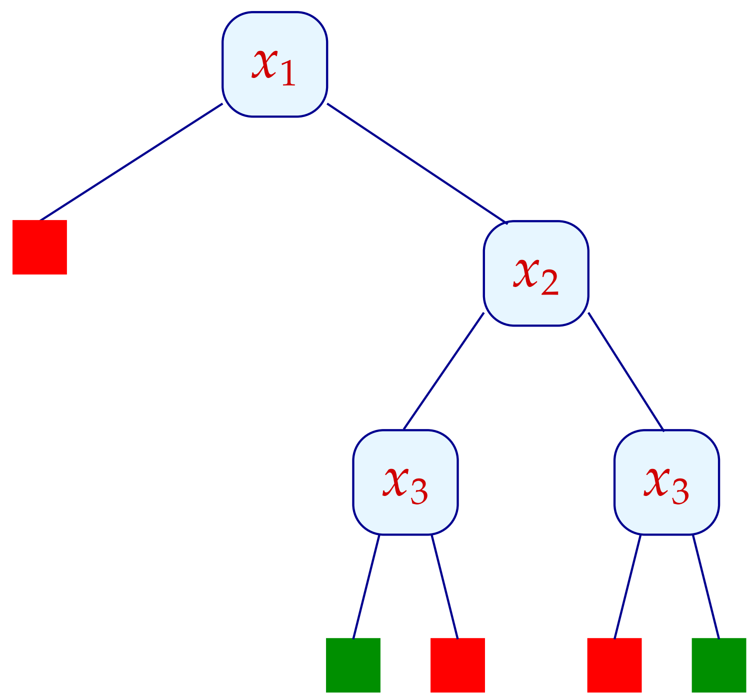
Idee (2):

- Entscheidungsbäume sind exponentiell groß :-(
- Oft sind jedoch viele Teilbäume isomorph :-)
- Isomorphe Teilbäume repräsentieren wir nur einmal ...



Idee (3):

- Knoten, deren Abfragen keine Rolle spielt, können ebenfalls eingespart werden ...



Diskussion:

- Die Repräsentation der Booleschen Funktion f ist eindeutig !

  $\Longrightarrow$

  Gleichheit von Funktionen ist effizient entscheidbar !!

- Damit die Darstellung brauchbar ist, sollte sie die grundlegenden Operationen: $\wedge, \vee, \neg, \Rightarrow, \exists\, x_j$ unterstützen ...

$$
\begin{array}{lcl}
[b_1 \wedge b_2]_k & = & b_1 \wedge b_2 \\
[f \wedge g]_{i-1} & = & \textbf{fn}\ x_i \ \Rightarrow\ \textbf{if}\ x_i\ \textbf{then}\ [f\,1 \wedge g\,1]_i \\
 & & \qquad\qquad\quad \textbf{else}\ [f\,0 \wedge g\,0]_i \\
 & & // \quad \text{analog für die anderen Operatoren}
\end{array}
$$

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

- Operationen werden bottom-up ausgeführt.
- Wurzelknoten bereits konstruierter Teilgraphen sind in einer Unique-Table abgelegt

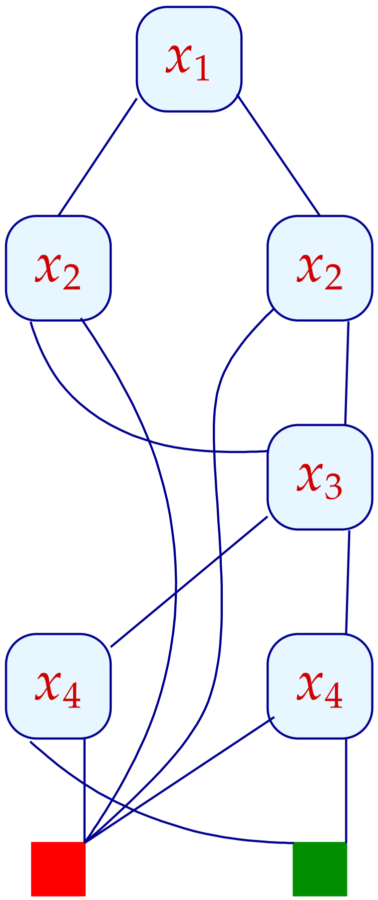
  $\Longrightarrow$

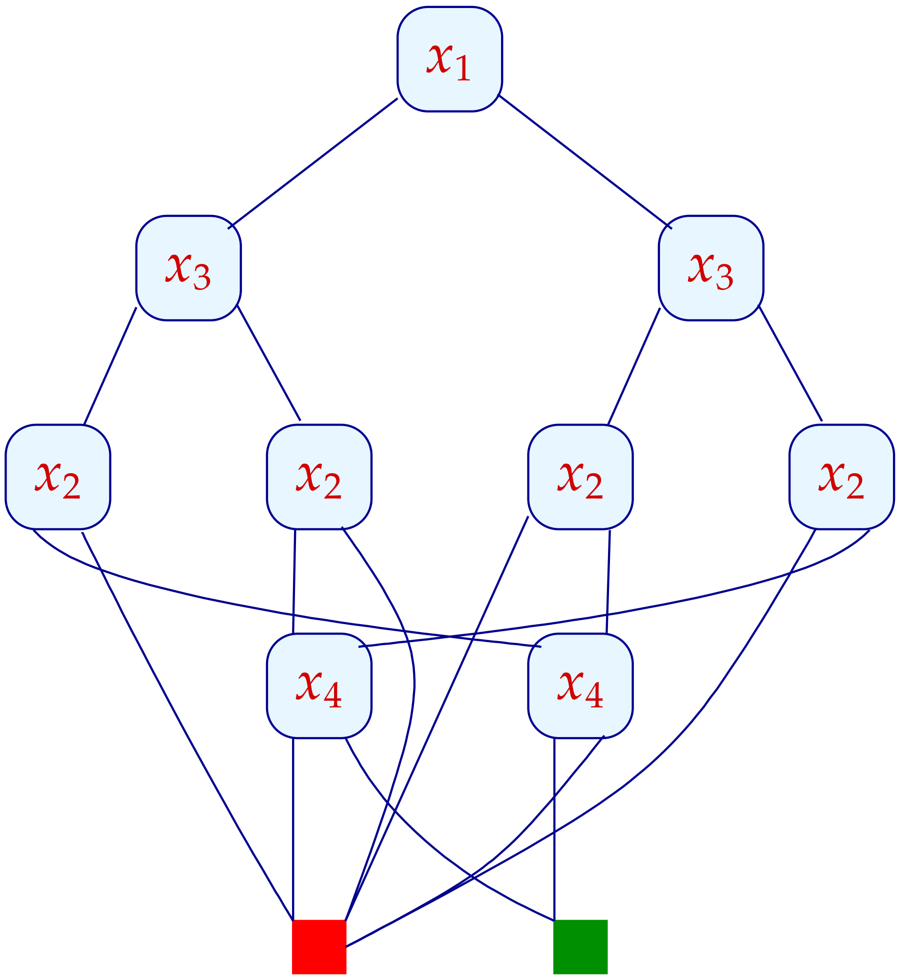
  Test auf Isomorphie in konstanter Zeit möglich !
- Operationen sind damit polynomiell in der Größe der Eingabe-BDDs :-)

Diskussion:

- BDDs wurden ursprünglich entwickelt, um Schaltkreise zu verifizieren.
- Heute werden sie auch zur Verifikation von Software eingesetzt ...
- Ein Systemzustand wird durch eine Folge von Bits repräsentiert.
- Ein BDD beschreibt dann die Menge aller erreichbaren Systemzustände.
- Achtung: Wiederholte Anwendung Boolescher Operationen kann die Größe dramatisch vergrößern !
- Die Variablenanordung ist von entscheidender Bedeutung ...

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





Diskussion (2):

- Betrachten wir allgemein die Funktion

$$(x_1 \leftrightarrow x_2) \wedge \ldots \wedge (x_{2n-1} \leftrightarrow x_{2n})$$

Bzgl. der Variablenanordnung:

$$x_1 < x_2 < \ldots < x_{2n}$$

hat der BDD $3n$ innere Knoten.

Bzgl. der Variablenanordnung:

$$x_1 < x_3 < \ldots < x_{2n-1} < x_2 < x_4 < \ldots < x_{2n}$$

hat der BDD mehr als 2^n innere Knoten !!

- Ähnliches gilt, wenn man Addition durch BDDs implementiert.

Diskussion (3):

- Nicht alle Booleschen Funktionen haben kleine BDDs :-(
- Schwierige Funktionen:
  - Multiplikation;
  - indirekte Adressierung ...

$\Longrightarrow$ datenintensive Programme so nicht analysierbar :-(

Ausblick: Andere Programmeigenschaften

Freeness: Ist X_i stets ungebunden ?

$\Longrightarrow$

Ist X_i stets gebunden, liefert Indexing für X_i eine genaue Fallunterscheidung :-)

Pair Sharing: Sind X_i, X_j an Terme t_i, t_j gebunden mit

$$Vars(t_i) \cap Vars(t_j) \neq \emptyset \quad ?$$

$\Longrightarrow$

Literale ohne Sharing können parallel ausgeführt werden :-)

Achtung:

Beide Analysen profitieren von Groundness !