

## 6. Entscheidungsprozeduren

Entscheidungsprozedur: Prozedur zur Lösung des Entscheidungsproblem in einer (entscheidbaren) Theorie.

Warum Entscheidungsprozeduren?

- Deduktionen in häufig benutzten, entscheidbaren Theorien werden automatisch ausgeführt;  
Entscheidungsprozeduren als “*black boxes*”
- Lästige, langwierige und meistens uninteressant Routine-Manipulationen werden von der Maschine übernommen; der Benutzer kann sich auf die wesentlich(er)en Aspekte eines Beweises konzentrieren.

Früher haben einige Beweiserbauer eine puristische Haltung eingenommen, wenn ihre Philosophie Entscheidungsprozeduren nicht zuließ (Beispiel: HOL). Heute enthalten alle praktisch eingesetzten Beweiser Entscheidungsprozeduren.

## Entscheidungsprozeduren (2)

Typische entscheidbare Theorien:

- boolesche Ausdrücke (Aussagenlogik)
- Lineare Arithmetik (Presburger-Arithmetik) mit Ordnungsrelation
- *Allgemeine* Gleichungstheorie: Gleichungen mit uninterpretierten Funktionssymbolen  
(im Gegensatz zu durch Gleichungen spezifizierte Theorien)
- einfache Datenstrukturen (bzw. bestimmte Teiltheorien dazu):
  - Arrays, Records
  - einfache induktive Datenstrukturen mit Konstruktoren und Selektoren (Listen, Bäume, . . . ).

## Entscheidungsprozeduren (3)

Probleme und Beschänkungen:

- Viele für Anwendungen interessante Theorien sind nicht entscheidbar.  
Beispiel: nicht-lineare Arithmetik  
Partielle Lösung: Entwicklung von *unvollständigen* Verfahren, die einfachere Fälle erkennen und vereinfachen.
- Die meisten Entscheidungsprobleme sind inhärent komplex, d.h. effiziente Entscheidungsprozeduren kann es im allgemeinen nicht geben.  
~> Unterschiedliche Realisierungen von Entscheidungsprozeduren, die jeweils für eingeschränkte Anwendungsbereiche hinreichend effizient sind (Beispiel: Aussagenlogik).
- Entscheidungsprobleme werden theoretisch meist in Isolation untersucht, aber Entscheidungsprozeduren werden i.a. zusammen mit anderen Beweisverfahren (und anderen Entscheidungsprozeduren) eingesetzt und müssen mit diesen harmonieren.  
~> Problem der Kombination von Entscheidungsprozeduren und deren Integration in Beweissysteme.

## Entscheidungsprozeduren (4)

*Implementierung*: z.B. durch

- kanonische Termersetzungssysteme
- spezielle Algorithmen

Z.B. Array-Theorie:

Sorten:

$I$  – Index-Typ

$E$  – Element-Typ

$Arr$  – Arrays über  $I$  und  $E$

Signatur:

$update : Arr \times I \times E \rightarrow Arr$

$select : Arr \times I \rightarrow E$

Axiome:

$select(update(A, i, e), j) = [\mathbf{if } i=j \mathbf{ then } e \mathbf{ else } select(A, j)]$

# Kongruenz-Abschluß

(engl. *congruence closure*)

Ausgangspunkt: gerichteter markierter Graph mit markierten Kanten; zwischen zwei Knoten können mehrere Kanten existieren.

Für eine Knotenmenge  $V$  und  $v \in V$  bezeichne

$l(v)$  die Marke von  $v$ ,

$\delta(v)$  die Anzahl der Nachfolger von  $v$ ,

$v[i]$  den  $i$ -ten Nachfolger-Knoten von  $v$  ( $1 \leq i \leq \delta(v)$ ).

$R$  sei eine Relation auf  $V$ .

Knoten  $u, v \in V$  heißen *kongruent unter  $R$* , falls sie gleich markiert sind, die gleiche Anzahl von Nachfolgern haben, und die jeweils entsprechenden Nachfolger in der Relation  $R$  stehen:

$$u \approx_R v \iff l(u) = l(v) \wedge \delta(u) = \delta(v) \wedge \\ \forall i : \text{Nat. } 1 \leq i \leq \delta(u) \Rightarrow (u[i], v[i]) \in R$$

## Kongruenz-Abschluß (2)

$R$  ist abgeschlossen unter Kongruenz, falls  $(u, v) \in R$  gilt für alle Paare  $u, v$ , die kongruent unter  $R$  sind, d.h. falls gilt

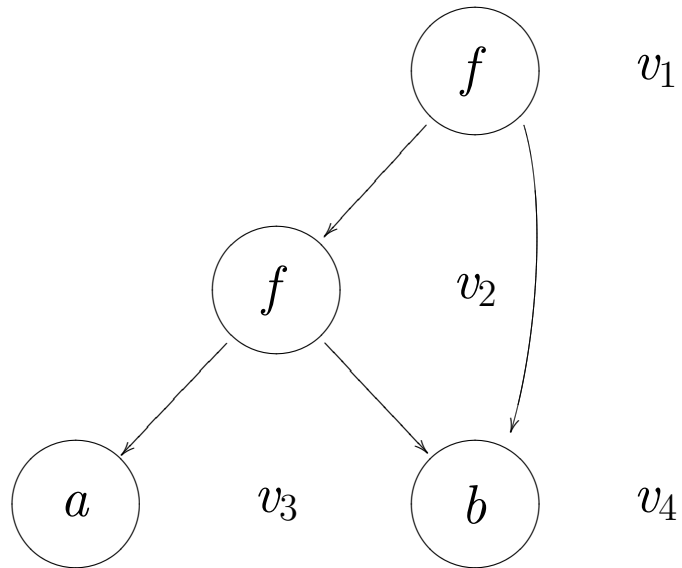
$$u \approx_R v \Rightarrow (u, v) \in R$$

Der Kongruenzabschluß  $\hat{R}$  von  $R$  ist die kleinste Äquivalenzrelation, die  $R$  enthält und unter Kongruenz abgeschlossen ist.

Intendierte Interpretation: Graph stellt Terme eindeutig dar (derselbe Term taucht höchstens einmal auf); die Ausgangsrelation  $R$  stellt eine Menge von Gleichungen dar; der Kongruenzabschluß von  $R$  repräsentiert alle Gleichungen, die sich aus  $R$  ableiten lassen.

Konstruktion des Kongruenz-Abschlusses wird benutzt als Grundlage einer Entscheidungsprozedur für quantorenfreie Gleichungstheorie mit uninterpretierten Funktionssymbolen (d.h. implizit universell quantifizierte Gleichungen zwischen Termen): einziges Prädikat ist die Gleichheit (=).

## Beispiel 1:



Gegebene Gleichung:  $f(a, b) = a$   
ergibt Ausgangsrelation

$$R := \{(v_2, v_3)\}$$

d.h.  $v_1, v_2$  sind unter  $R$  kongruent.

Kongruenzabschluß  $\hat{R}$  von  $R$ :

Äquivalenzklassen  $\{v_1, v_2, v_3\}, \{v_4\}$  mit

$$v_1 : f(f(a, b), b) \text{ usw.}$$

$v_1 \approx_{\hat{R}} v_3$  entspricht der Aussage

$$f(f(a, b), b) = a$$

## Beispiel 2:

Ausgangsrelation:

$$R := \{(v_1, v_6), (v_3, v_6)\}$$

Konstruktion des Kongruenzabschlusses  $\hat{R}$ :

$$(1) v_2, v_5 \text{ kongruent unter } \hat{R} \rightsquigarrow (v_2, v_5) \in \hat{R}$$

$$(2) v_1, v_4 \text{ kongruent unter } \hat{R} \rightsquigarrow (v_1, v_4) \in \hat{R}$$

Wegen Äquivalenzeigenschaft:

$$(3) \rightsquigarrow (v_4, v_6) \in \hat{R}$$

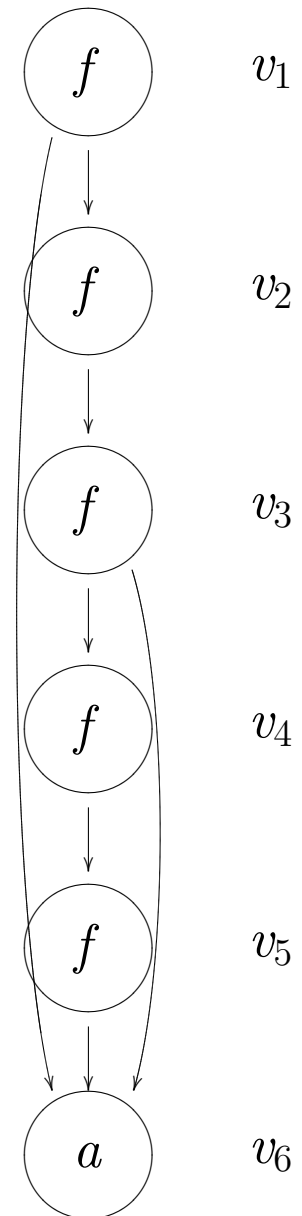
$$(4) \rightsquigarrow (v_3, v_5) \in \hat{R}$$

$\rightsquigarrow$  alle Knoten sind äquivalent in  $\hat{R}$

Logisch:

$$f^3(a) = a \wedge f^5(a) = a \Rightarrow f(a) = a$$

Frage: Kann  $f(a) = a$  aus  $f^3(a) = a$  und  $f^5(a) = a$  mit Hilfe von Termersetzung geschlossen werden?



## Algorithmus zur Berechnung des Kongruenz-Abschlusses:

Die Äquivalenz-Relation wird durch eine Partition (disjunkte Zerlegung) der Knotenmenge  $V$  in Äquivalenz-Klassen repräsentiert.

Vorgegebene Prozeduren:

$Union(u, v)$ : Vereinigung der Äquivalenzklassen von  $u$  und  $v$

$Find(u)$ : Auffinden der Äquivalenzklasse von  $u$  – gibt eindeutigen “Namen” der Äquivalenzklasse zurück

$Merge(u, v)$ : Füge  $(u, v)$  zur Relation hinzu und schließe unter Kongruenz ab.

Falls  $Find(u) = Find(v)$ , fertig.

$P_u$  bezeichne jeweils die Menge aller Vorgänger der zu  $u$  äquivalenten Knoten; entsprechend  $P_v$ .

Rufe  $Union(u, v)$  auf.

Für jedes Paar  $(x, y)$  mit  $x \in P_u$  und  $y \in P_v$ , falls  $Find(x) \neq Find(y)$  und  $Congruent(x, y)$ , rufe  $Union(x, y)$  auf.

$Congruent(x, y)$ : Test, ob  $x$  und  $y$  in der derzeitigen Relation kongruent sind.

Falls  $\delta(x) \neq \delta(y)$ , Ergebnis *Falsch*.

Für alle  $1 \leq i \leq \delta(x)$ , falls immer  $Find(x[i]) = Find(y[i])$ , Ergebnis *Wahr*.

Andernfalls Ergebnis *Falsch*.

Gültigkeit einer Formel ist äquivalent zur Unerfüllbarkeit der negierten Formel in DNF, damit reduziert auf die (Un-)Erfüllbarkeit von Konjunktionen von Literalen.

Entscheidungsprozedur: Erfüllbarkeit einer Konjunktion  $K$  von Gleichungen und Ungleichungen (d.h. negierten Gleichungen)

$$t_1 = u_1 \wedge \dots \wedge t_p = u_p \wedge r_1 \neq s_1 \wedge \dots \wedge r_q \neq s_q$$

1. Konstruiere einen Graphen  $G$ , der eindeutig alle Terme repräsentiert, die in  $K$  vorkommen.

Für einen Term  $t$  bezeichne  $\tau(t)$  den ihn repräsentierenden Knoten.

$R$  ist zu Beginn die Identitätsrelation.

2. Für  $i \leq p$ , rufe  $Merge(\tau(t_i), \tau(u_i))$  auf.

3.  $K$  ist unerfüllbar, falls  $\tau(r_j)$  und  $\tau(s_j)$  äquivalent sind für ein  $j \leq q$ .

4. Andernfalls ist  $K$  erfüllbar.

# Erweiterung für induktive Datenstrukturen

Beispiel: quantorenfreie Theorie der Binärbäume als Erweiterung der obigen Gleichungstheorie:

Funktionssymbole  $comp$ ,  $left$ ,  $right$

Prädikatssymbolen  $atom?$ ,  $comp?$

Axiome:

$$left(comp(x, y)) = x$$

$$right(comp(x, y)) = y$$

$$comp?(x) \Rightarrow comp(left(x), right(x)) = x$$

$$comp?(comp(x, y))$$

Beispiel eines Theorems der Theorie:

$$left(x) = left(y) \wedge right(x) = right(y) \wedge comp?(x) \wedge comp?(y)$$

$$\Rightarrow f(x) = f(y)$$

Bemerkung: Negierte Vorkommen von  $atom?(t)$  bzw.  $comp?(t)$  können immer vermieden werden durch Verwendung des jeweils komplementären Prädikats.

## Erweiterung für induktive Datenstrukturen (2)

*Entscheidungsprozedur* für Erfüllbarkeit von Konjunktionen der Form

$$atom?(u_1) \wedge \dots \wedge atom?(u_q) \wedge \\ v_1 = w_1 \wedge \dots \wedge v_r = w_r \wedge x_1 \neq y_1 \dots \wedge x_s \neq y_s$$

Terme können *comp*, *left*, *right* und uninterpretierte Funktionssymbole enthalten.

1. Konstruiere Graphen  $G$  für alle Terme:

$$Merge(\tau(v_i), \tau(w_i)) \quad \text{für alle } i \leq r.$$

2. Für jeden Knoten, der einen Term  $comp(x, y)$  repräsentiert, füge Knoten für  $left(comp(x, y))$  und  $right(comp(x, y))$  hinzu, *Merge* mit  $x$  bzw.  $y$ .

3. Teste auf (Un-)Erfüllbarkeit der Ungleichungen.

# OBDDs

OBDD – “*Ordered Binary Decision Diagrams*”

“Geordnete binäre Entscheidungsdiagramme”

Verfahren zum schnellen Entscheiden der Gültigkeit von aussagenlogischen Formeln

*Binäres Entscheidungsdiagramm*: in der einfachsten Form ein binärer *Entscheidungsbaum*, in dem bei jedem (inneren) Knoten eine binäre Entscheidung bezüglich eines aussagenlogischen Symbols  $x$  (einer “Variablen”) getroffen wird, d.h. verzweigt mit “ $x$  wahr” bzw. “ $x$  falsch”.

*Geordnet*: die aussagenlogischen Symbole sind (total) geordnet bezgl. der Reihenfolge, in der sie bei Entscheidungen auftreten.

OBDD-Verfahren sind seit einigen Jahren sehr populär als Methoden zum Nachweis der Äquivalenz von aussagenlogischen Ausdrücken, wie sie insbesondere bei der Modellierung von Hardware-Komponenten anfallen. In diesem Bereich sind die Verfahren oft um mehrere Größenordnungen schneller als herkömmliche Verfahren.

Grundlage vieler Systeme zur Modellüberprüfung (*model checking*) – kommt später.

## OBDDs (2)

Grundidee:

- Logische Formel wird umgeformt in eine äquivalente Formel in *bedingter Normalform*: alle logischen Verknüpfungen werden durch (Kombinationen von) **if\_then\_else** ersetzt.

Bem.: Bedingte Normalformen sind nicht eindeutig.

- Bei der Umformung werden die zu testenden Symbole in die vorgegebene Reihenfolge gebracht.

Dadurch wird eine *kanonische*, eindeutige bedingte Normalform erreicht.

Der Test, ob zwei aussagenlogische Formeln äquivalent sind, wird dadurch reduziert auf den Test, ob ihre kanonischen Normalformen identisch sind.

- Die Umformung in bedingte Normalform kann zu exponentiellem Wachstum der Größe führen.

Durch Anwendung verschiedener Codierungstechniken kann die Effizienz enorm gesteigert werden.

## OBDDs (3)

**Umformung** in bedingte Normalform:

mit  $if(a, b, c)$  als Kurzform für **if**  $a$  **then**  $b$  **else**  $c$

Transformationen:

$$\begin{aligned} A \wedge B &\rightarrow if(A, B, \mathbf{F}) \\ A \vee B &\rightarrow if(A, \mathbf{W}, B) \\ \neg A &\rightarrow if(A, \mathbf{F}, \mathbf{W}) \\ A \Rightarrow B &\rightarrow if(A, B, \mathbf{W}) \\ A \Leftrightarrow B &\rightarrow if(A, B, if(B, \mathbf{F}, \mathbf{W})) \\ A \oplus B &\rightarrow if(A, if(B, \mathbf{F}, \mathbf{W}), B) \quad (\text{“exkl. Oder”}) \end{aligned}$$

- (1)  $if(\mathbf{W}, A, B) \rightarrow A$
- (2)  $if(\mathbf{F}, A, B) \rightarrow B$
- (3)  $if(A, B, B) \rightarrow B$
- (4)  $if(if(A, B, C), D, E) \rightarrow if(A, if(B, D, E), if(C, D, E))$
- (5)  $if(x, A, B) \rightarrow if(x, A[x \leftarrow \mathbf{W}], B[x \leftarrow \mathbf{F}])$
- (6)  $a \rightarrow if(a, \mathbf{W}, \mathbf{F})$  für Symbol  $a$  als Blatt

## OBDDs (4)

*Bedingte Normalformen:*

- Die Konstanten **W** oder **F**, oder
- ein Term der Form  $if(x, a, b)$  mit
  - $x$  ein Symbol ungleich einer der Konstanten **W** oder **F** (Regeln (1), (2), (4)),
  - $a$  und  $b$  sind ungleich (Regel (3)),  
in bedingter Normalform (rekursiver Prozeß bzw. Regel (6) im letzten Schritt),  
und enthalten kein Vorkommen von  $x$  (Regel (5)).

## OBDDs (5): Beispiel einer Normalisierung

$$\begin{aligned} & if(if(A, C, B), if(A, B, \mathbf{W}), \mathbf{W}) \\ = & if(A, if(C, if(A, B, \mathbf{W}), \mathbf{W}), if(B, if(A, B, \mathbf{W}), \mathbf{W})) & (4) \\ = & if(A, if(C, if(\mathbf{W}, B, \mathbf{W}), \mathbf{W}), if(B, if(\mathbf{F}, B, \mathbf{W}), \mathbf{W})) & (5) \\ = & if(A, if(C, B, \mathbf{W}), if(B, \mathbf{W}, \mathbf{W})) & (1), (2) \\ = & if(A, if(C, B, \mathbf{W}), \mathbf{W}) & (3) \\ = & if(A, if(C, if(B, \mathbf{W}, \mathbf{F}), \mathbf{W}), \mathbf{W}) & (6) \end{aligned}$$

Bemerkung: eine Tautologie wird durch diesen Prozeß zu  $\mathbf{W}$  reduziert  
 $\rightsquigarrow$  Entscheidungsprozedur für Tautologien.

## OBDDs (6)

*Kanonische Normalform:*

Bedingte Normalform, in der die Folge von Symbolen in Testpositionen entlang eines Pfads von der “Wurzel” zu einem “Blatt” der vorgegebenen Ordnung auf der Menge der Symbole genügt.

Jeder aussagenlogische Ausdruck kann in eine solche kanonische Normalform gebracht werden durch “Liften” des kleinsten Symbols  $x$  aus dem Ausdruck  $a$ , d.h. eine Transformation

$$A \rightarrow if(x, A, A)$$

und anschließender Reduktion der Ausdrücke  $a$  nach Regel (5) und weiterer Kanonisierung.

*Beispiel:*  $(A \oplus C) \oplus B$

Ordnung der Symbole ist alphabetisch. Kanonische bedingte Form:

$$if(A, if(B, if(C, \mathbf{W}, \mathbf{F}), if(C, \mathbf{F}, \mathbf{W})), \\ if(B, if(C, \mathbf{F}, \mathbf{W}), if(C, \mathbf{W}, \mathbf{F})),$$

## OBDDs (7): Effizienz-Steigerung

Einfacher Algorithmus für das Erzeugen der kanonischen Form: rekursiver Abstieg in den Ausdruck, Verknüpfung der kanonisierten Argumente entsprechend des Operationssymbols.

Dieser “naive” Algorithmus ist zu ineffizient. Verschiedene Implementierungstechniken können die Effizienz ganz erheblich steigern.

### 1. “merge sort”

Argumente  $x = if(vx, tx, fx)$  und  $y = if(vy, ty, fy)$  in kanonisierter Form; zu berechnen ist die kanonische Form von  $op(x, y)$ .

3 Fälle sind zu unterscheiden:

$$(a) \quad vx = vy: \quad op(x, y) = if(vx, op(tx, ty), op(fx, fy))$$

$\rightsquigarrow$   $op$  wird rekursiv in Teilausdrücke propagiert.

$$(b) \quad vx < vy: \quad op(x, y) = if(vx, op(tx, y), op(fx, y))$$

Wegen Ordnung kann  $vx$  in  $y$  nicht auftreten; es reicht, die Teilausdrücke zu kanonisieren.

$$(c) \quad vy < vx: \quad \text{symmetrisch zu (b)}$$

## OBDDs (8)

Beispielableitung für  $(A \oplus C) \oplus B$ :

Nach Kanonisierung der Argumente:

$$if(A, id(C, \mathbf{F}, \mathbf{W}), id(C, \mathbf{W}, \mathbf{F})) \oplus if(B, \mathbf{W}, \mathbf{F})$$

Lifting  $A$ :  $\rightsquigarrow$

$$if(A, id(C, \mathbf{F}, \mathbf{W}) \oplus if(B, \mathbf{W}, \mathbf{F}), id(C, \mathbf{W}, \mathbf{F}) \oplus if(B, \mathbf{W}, \mathbf{F}))$$

Operation  $\oplus$  wird durch den Ausdruck propagiert, mit entsprechenden Vereinfachungen, zum Ergebnis wie oben.

## OBDDs (9):

2. Eindeutige und einmalige Darstellung von (Unter-)Ausdrücken:

In der graphischen Darstellung werden identische Teil-Bäume (-Diagramme) verschmolzen, indem alle hineingehenden Kanten auf die Wurzel eines Exemplars zeigen und die anderen gelöscht werden; anfangend mit den Terminalknoten **W** und **F**.

Benutzung einer *Hash-Funktion* zur Unterstützung der eindeutigen Darstellung:

dient unter anderem der effizienten Realisierung der Regel (3): Wenn ein If-Ausdruck  $if(vx, tx, fx)$  generiert wird, muß jeweils geprüft werden, ob  $tx = ty$ .

- Jedem If-Ausdruck und jeder Operation wird ein eindeutiger Wert zugeordnet.
- Aus dem Tripel der Werte für  $op$ ,  $tx$  und  $ty$  wird ein Hash-Index berechnet.
- Hash-Konflikte werden in der üblichen Weise (z.B. über eine Tabelle) aufgelöst.
- Test auf Gleichheit von Teilausdrücken ist damit reduziert auf den direkten Vergleich der eindeutigen, den Ausdrücken zugeordneten Werten.

## OBDDs (10):

3. Ausnutzung einer *Memo-Funktion*: Das Ergebnis der Kanonisierung eines Ausdrucks (mit dem eindeutigen, dem kanonisierten Ausdruck zugeordneten Wert) wird zusammen mit dem Ausdruck in einer Hash-Tabelle gespeichert; damit kann wiederholte Kanonisierung desselben Ausdrucks vermieden werden.