MONA

Von schwacher MSO zum endlichen Automaten

Martin Emrich

Betreuer: Tim Priesnitz, Thomas Klöcker, Christian Klein

Proseminar "Logische Aspekte von XML" - SS 2003

Gert Smolka

PS-Lab, Uni Saarland

MONA – Beweissystem für schwache MSO

- Logikbasierte Programmiersprache
- Berechnet Tupel-DEA aus MSO-Formeln
- Anwendungsgebiete:
  - Protokollverifikation
  - Hardwareverifikation
  - Linguistik
  - Controllersysteme

Mona ist ein Beweissystem für schwache Monadische Logik zweiter Stufe.

Es implementiert eine Logikbasierte Programmiersprache, in der MSO-Formel n notiert werden.

In C/C++ geschrieben

Übersicht

- MSO
- Belegung als Matrix – Besonderheiten in MONA
- Ein Beispiel
- Berechnung und Optimierungskonzepte
  - Dreiwertige Semantik für MSO
  - BDDs als effiziente Automatendarstellung
- Berechnung des Beispiels

MSO

Individuen

$$t ::= 0 \mid p \mid t+1$$

Mengen

$$T ::= \emptyset \mid P \mid T \cap T \mid T \cup T \mid T \setminus T$$

Formeln

$$\begin{aligned} \phi ::= {} & t \leq t \mid t = t \mid T = T \mid t \in T \\ & \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \Rightarrow \phi \mid \phi \Leftrightarrow \phi \\ & \mid \exists p : \phi \mid \forall p : \phi \mid \exists P : \phi \mid \forall P : \phi \end{aligned}$$

[Kla98]

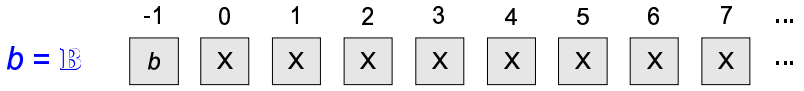
Mathematische Symbole, da geläufiger, MONA-Syntax findet sich im Handbuch.

Komplette Redundate Grammatik, viel Syntaktischer Zucker, z.B: Makros

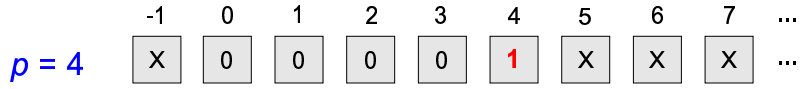
Darstellung von Belegungen

Besonderheiten bei MONA

Boole'sche Werte als erstes Zeichen



Individuen p als Menge P mit p=min(P)



$Shape(X) = 1^{max(|P|)}$

[KM01],[KMS00]

(Boolesche)

Im ersten Zeichen werden alle booleschen Variablen kodiert. Dies ermöglicht MONA, sie bei der Automatenkonstruktion frühzeitig zu berücksichtigen. --> Optimierung

(Individuen)

werden nicht als Singleton kodiert, sondern als nichtleere Menge mit dem Wert als kleinstem Element.

Mengen werden wie bekannt kodiert.

Beispiel

Formel:

$$x \in X \Rightarrow x1 \in Y$$

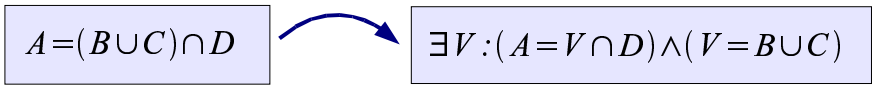
MONA-Programm:

```
var1 x; var2 X,Y;
x in X => x+1 in Y;
```

wenn x in X ist, so ist der Nachfolger von x in Y.

Syntaktische Vereinfachung

Terme 2. Ordung werden „aufgefaltet“



Entfernen redundanter Operationen

Kernsprache aus atomaren Formeln

$\Phi ::= \neg\Phi \mid \Phi\wedge\Phi \mid \exists X{:}\Phi$
$\mid X\subseteq X' \mid X=X'+1 \mid X=X' \setminus X''$

Auffaltung:

Erzeugen atomarer Automaten -> Vortrag von Jens

All -> Existenz

Probleme

- Individuen als nichtleere Mengen

$\Phi = p=2$ → $\Phi' = P=\{2\}$

Probleme

- Individuen als nichtleere Mengen

$\Phi = \neg(p=2)$  $\Phi' = \neg(P=\{2\})$

Nicht überall definiert - {} ?!?

Probleme

•Individuen als nichtleere Mengen

$\Phi = \neg(p{=}2)$ → $\Phi' = \neg(P{=}\{2\}) \wedge singleton(P)$

•MSO auf Strings: Beschränkung von Mengen

Konjunktion mit der Beschränkung bei jedem Auftreten von *p* führt zu größerem DEA

Individuen werden in MONA nicht als Singleton kodiert, sondern als nichtleere Menge... ist leicht effizienter.

Was ist mit P={} ? Phi ist nicht auf diese Mengen definiert, sondern nur auf Singletons bzw. nichtleere Mengen.

Konjunktion und anschliessende Minimierung

WS1S-R

Existenzquantifizierung

$\Phi = \exists X: \Phi' \wedge \texttt{singleton}(X)$ → $\Phi = \exists X \texttt{ where } \texttt{singleton}(X) : \Phi'$

(MONA: Implizit bei Individuenvariablen)

Φ nur definiert wenn $\texttt{singleton}(X)$ erfüllt ist

Einschränkungen explizit in der Syntax

$$\Phi = \exists X \texttt{ where } \rho : \Phi'$$

Anwendung z.B. für WS1S über Strings

[Kla99]

Bei M2L-Str werden Mengen auf {0…n} beschränkt

Dreiwertige Semantik für MSO

MONA: Einschränkungen durch dreiwertige Semantik

- $\mathbb{B}$ geliftet auf $\mathbb{B}_\perp$

Nicht alle Einschränkungen für Variablen in Φ erfüllt

Φ wertet zu $\perp$ aus

- Erweiterung des Automaten um *don't-care*-Zustände

[Kla99]

Lösung:

Anhebung der Semantik auf B_bottom. Intutition: !Einschränkung nicht erfüllt ->Bottom

Dreiwertige Semantik für MSO

$\boldsymbol{\rho}^*(P)$:= Konjunktion aller Einschränkungen für P

$[\Phi \wedge \Phi']^3 w = [\Phi]^3 w \wedge^3 [\Phi']^3 w$

$[\neg \Phi']^3 w = \neg^3 [\Phi]^3 w$

$[P \subseteq P']^3 w =$

$\bot$	wenn	$[(\boldsymbol{\rho}^*(P) \wedge \boldsymbol{\rho}^*(P'))]^3 w \neq 1$,
1	wenn	$w \models (P \subseteq P')$ und $[(\boldsymbol{\rho}^*(P) \wedge \boldsymbol{\rho}^*(P'))]^3 w = 1$
0	sonst	

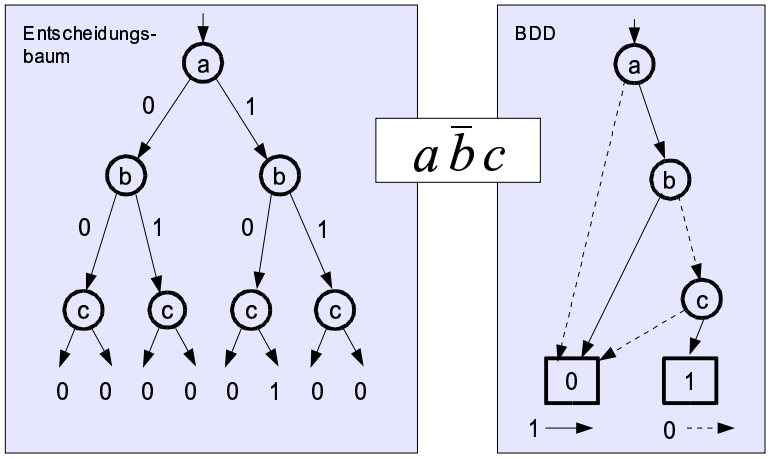
$\Phi = \exists P$ **where** $\rho : \Phi'$

Wertet zu $\bot$ aus wenn P Einschränkung ρ nicht erfüllt

[KM01]

BDD

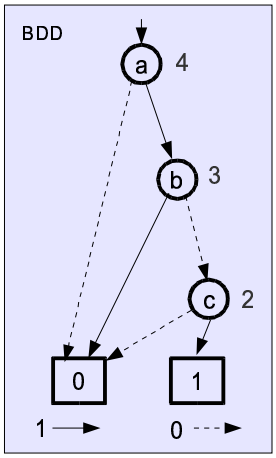
Gerichteter Graph statt Enscheidungsbaum



BDD – Definition

- DAG mit Wurzel
- Besteht aus
  - Endknoten {0,1}
  - Variablenknoten $\subset Var$
  - Übergangsfunktion γ: $\exists n \in \mathbb{N}$: $\gamma \in \{2..n\} \rightarrow Var \times \{0...n\} \times \{0...n\}$
- **geordneter BDD (OBDD)**
  - $\forall (n,(X,n_0,n_1)) \in \gamma : n>n_0 \wedge n>n_1$ Ordnung auf Variablenknoten

$\gamma = \{(4,(a,0,3)),(3,(b,2,0)),(2,(c,0,1))\}$



[Sm03]

Übergangsfunktion Gamma

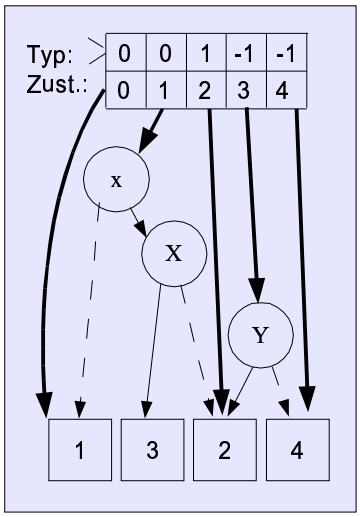
Start bei der Wurzel klar, Schaue in Übergangsfunktion bei [Anzahl Knoten], finde übergang bei n, dann nur noch suche ab n-1.

SMBDDs in MONA

- *shared, multi-terminal BDD*
- DEA kodiert als gerichter Graph
- Endknoten = DEA-Zustände
- nicht mehr azyklisch:
  1 Zyklus / Eingabezeichen
- geordnet nach Variablen

Hier:

- nur 7 Knoten, statt bis zu 40
  Einträgen in Übergangstabelle



[KM01]

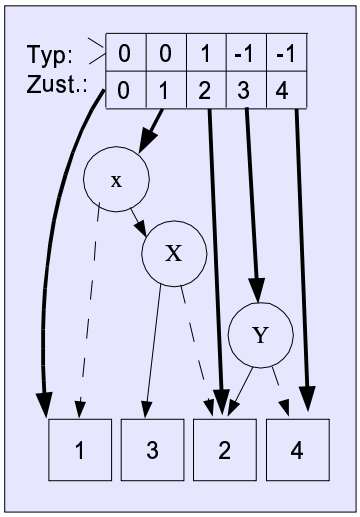
Beispiel im SMBDD

$x \in X \Rightarrow x1 \in Y$

```
var1 x; var2 X,Y;
x in X => x+1 in Y;
```

w=

```
x X001XXX⊥
X X001100⊥
Y X000111⊥
```



Beispiel mit BDD

0: don't care

1: accept

-1: reject

Nach 1 Zeichen noch don't care, da x ={}

nach x=„001“ =2 kein don't care,

->Menge mit kleinstem Element Notation

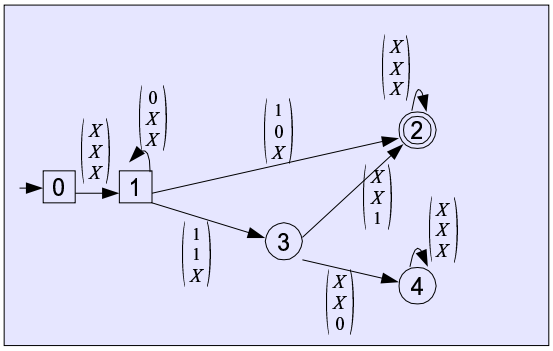
Beispiel im DEA

$x \in X \Rightarrow x1 \in Y$

```
var1 x; var2 X,Y;
x in X => x+1 in Y;
```

$w'=$

```
x X001XXX⊥
X X101100⊥
Y X000111⊥
```



Hier keine Erfüllung

Quellen

- [KM01] Nils Klarlund, Anders Møller.
  MONA Version 1.4 Users Manual.
  BRICS Notes Series NS-01-1, Dept. of Computer Science, University of Aarhus, Jan. 2001
- [KMS00] Nils Klarlund, Anders Møller, Michael I. Schwartzbach.
  MONA Implementation Secrets
  In *Fifth International Conference on Implementation and Application of Automata, CIAA'00*, 2000
- [Kla99] Nils Klarlund.
  A Theory of restrictions for logics and automata
  In *Computer Aided Verification, CAV'99*, volume 1633 of LNCS, 1999
- [Kla98] Nils Klarlund.
  Mona & Fido: The Logic-Automaton Connection in Practice
  In *Computer Science Logic, CSL'97*, volume 1414 of LNCS, 1998
- [Sm03] Gert Smolka.
  Einführung in die computationale Logik
  Skript zur Vorlesung, PS Lab, Universität des Saarlandes, SS 2003

MONA Download unter `http://www.brics.dk/mona`