

# Formale Methoden und Werkzeuge

## Vorlesung

### Wintersemester 2024, 25

Johannes Waldmann, HTWK Leipzig

29. Oktober 2025

– Typeset by FoilTeX –

# Die Übung vor der ersten Vorlesung

## Organisation

- Folien: <https://www.imn.htwk-leipzig.de/~waldmann/lehre.html>
- Quelltexte, Diskussionen Hausaufgaben: <https://gitlab.dit.htwk-leipzig.de/johannes.waldmann/fmw-ws25>
- Software: ghc, cabal; minisat, ersatz; z3, hasmt; Agda
- nachfolgend: Beispiele für Hausaufgaben (Zweck: Wiederholung Prädikaten- und Aussagenlogik)

– Typeset by FoilTeX –

1

## Hausaufgaben

WS 25: 1b, 5, 4a, 2.

1. Pierluigi Crescenzi, Viggo Kann *A compendium of NP optimization problems*  
<https://web.archive.org/web/20200227195925/http://www.nada.kth.se/~viggo/problemlist/compendium.html>

Beispiele:

- (a) Minimum File Transfer Scheduling (node195).  
(b) Minimum Dynamic Storage Allocation (node163).

Erläutern Sie die Spezifikation an einem Beispiel. Geben Sie eine lösbare Instanz sowie dafür zwei Lösungen mit unterschiedlichem Maß an.

– Typeset by FoilTeX –

2

2. Aufgabe: formalisieren Sie *Math Magic* Februar 2007.

Was ist dabei für Springer und König einfacher als für Dame, Läufer, Turm?

(allgemein für Math Magic: offene Fragen <https://erich-friedman.github.io/mathmagic/unsolved.html>)

3. Aufgabe: formalisieren Sie Wolkenkratzer oder Towers (was ist der Unterschied?)

<https://www.janko.at/Raetsel/Wolkenkratzer>  
<https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/towers.html>

- $B = \{0, 1, \dots, n-1\}$ , Unbekannte:  $h_{x,y} \in B$  für  $x, y \in B$
- Constraint für eine Zeile  $x$ : (entspr. für Spalte)

– Typeset by FoilTeX –

3

$\bigvee_{p \in \text{Permutationen}(0, \dots, n-1), p \text{ kompatibel mit Vorgaben}} \bigwedge_{y \in \{0, \dots, n-1\}} (h_{x,y} = p(y))$

Bsp:  $n = 4$ , Vorgabe links 2, rechts 1, kompatibel sind  $[0, 2, 1, 3], [2, 0, 1, 3], [2, 1, 0, 3], [2, 1, 0, 3]$ .

- diese Formel wird exponentiell groß (wg. Anzahl Permutationen),  
Folge-Aufgabe: *geht das auch polynomiell?*
- Geben Sie eine Aufgabenstellung der Größe  $w \times w$  an mit  $4 \cdot w$  Vorgaben (d.h., *alle* Vorgaben), die mehr als eine Lösung hat.  
... für  $w = 4$ , für größere  $w$  (einige, alle)
- (offene Frage?) Geben Sie eine eindeutig lösbare Instanz mit möglichst wenigen Vorgaben an. (Sind  $w - 1$  Vorgaben für  $w \times w$  immer möglich?)

– Typeset by FoilTeX –

4

4. Modellierung von Puzzles (aus Tatham-Collection)

- (a) geben Sie ein Modell an für *Pegs* (Solitaire). Hinweise:
- Zustand = Boolesche Matrix,
  - Schritt = Relation zwischen Matrizen,
  - Lösung = Schrittfolge ( $\Rightarrow$  Zustandsfolge). Wieviele Schritte?
- (b) Modell für *Unruly*. (keine Zustandsfolge, sondern direkt die Lösung. Welches sind die Unbekannten, welches sind ihre Beziehungen, untereinander und zur Vorgabe)
- (c) modellieren Sie *Untangle*

Vergleichen Sie mit den tatsächlichen Quelltexten

<https://git.tartarus.org/?p=simon/puzzles.git>

– Typeset by FoilTeX –

5

5. Constraint für monotone kompatible Bewertungsfunktion:

- lösen Sie mit Z3 (ist im Pool installiert, vgl. <https://www.imn.htwk-leipzig.de/~waldmann/etc/pool/>)
- eine *kleinste* Lösung finden (Summe von  $P, Q, R, S$  möglichst klein) — dafür Assert(s) hinzufügen.
- Abstieg der so gefundenen Bewertungsfunktion nachrechnen für  $abab \rightarrow baab \rightarrow baba \rightarrow bbaa$
- gibt diese Bewertungsfunktion die maximale Schrittzahl genau wieder? (nein)
- Folge-Aufgabe: entspr. Constraint-System für Bewertungsfunktion für  $ab \rightarrow bba$  aufstellen und lösen.

– Typeset by FoilTeX –

6

– Typeset by FoilTeX –

7

# Einleitung

## Formale Methoden und Werkzeuge ...

- ... zur Spezifikation, Analyse (Verifikation), Synthese von Hard- und Softwaresystemen.
  - für (geplantes) System  $S$  (Bsp: CPU, Programm) *formales Modell*  $M$  angeben (Bsp: logische Schaltung, endl. Automat)
  - gewünschte System-Eigenschaft  $E$  angeben (Bsp: Boolesche Formel, reguläre Sprache)
  - Analyse: mit Werkzeug(unterstützung) feststellen, ob das Modell die Eigenschaft hat (Bsp:  $M \Rightarrow E$ ,  $M \subseteq E$ )
  - Synthese (*Constraint-Programmierung*): Werkzeug konstruiert aus Constraint  $E$  ein passendes Modell  $M$

– Typeset by FoilTeX –

8

# Die Rolle der Abstraktion

- Schritt von System  $S$  zu Modell  $M$  ist *Abstraktion* (es werden Details ignoriert), das ist
  - teils *nützlich* (verbessert die Übersicht, Einsicht),
  - teils *notwendig* (nur wenn  $M \subseteq E$  entscheidbar oder  $M$  aus  $E$  berechenbar, kann überhaupt ein Werkzeug zur Analyse bzw. Synthese implementiert werden)
- in den einfachen Beispielen (besonders am Anfang der VL) beginnen wir die Betrachtung bereits bei  $M$  (damit wir sehen, welche Theorien angewendet werden sollen)
- das Abstrahieren muß aber auch geübt werden, es ist aber eine anwendungsspezifische Kunst

– Typeset by FoilTeX –

9

## Industrielle Anwendungen von FMW

- Verifikation von Schaltkreisen (*bevor* man diese tatsächlich produziert)  
 $F = \text{S-Implementierung}(x) \neq \text{S-Spezifikation}(x)$   
wenn  $F$  unerfüllbar ( $\neg \exists x. F$ ), dann ist Impl. korrekt
- Verifikation von Software durch *model checking*:  
Programmzustände abstrahieren durch Zustandsprädikate, Programmabläufe durch endliche Automaten.  
z. B. Thomas Ball et al. 2004: *Static Driver Verifier*  
<https://www.microsoft.com/en-us/research/project/slam/publications/>  
benutzt Constraint-Solver Z3 (Nikolaj Björner et al., 2007–) <https://github.com/Z3Prover/z3/wiki>

– Typeset by FoilTeX –

10

## Industrielle Anwendungen von FMW

- automatische Analyse des Ressourcenverbrauchs von Programmen
  - Termination (jede Rechnung hält)
  - Komplexität (... nach  $O(n^2)$  Schritten)
- mittels *Bewertungen* von Programmezuständen:
  - $W : \text{Zustandsmenge} \rightarrow \mathbb{N}$
  - wenn  $z_1 \rightarrow z_2$ , dann  $W(z_1) > W(z_2)$ .
- Parameter der Bewertung werden durch Constraint-System beschrieben.
- vgl. Carsten Fuhs: *Automated Termination Analysis...*, Intl. School on Rewriting, 2022 <https://viam.science.tsu.ge/clas2022/isr/termination.html>

– Typeset by FoilTeX –

11

## Anwendung: Polynom-Interpretationen

- Berechnungsmodell: Wortersetzung ( $\approx$  Turingmaschine)
- Programm:  $ab \rightarrow ba$  ( $\approx$  Bubble-Sort)  
Beispiel-Rechnung:  $abab \rightarrow baab \rightarrow baba \rightarrow bbaa$
- Bewertung  $W$  durch *Interpretation*: lineare Funktionen  
 $f_a(x) = Px + Q$ ,  $f_b(x) = Rx + S$  mit  $P, Q, R, S \in \mathbb{N}$   
 $W(abab) = f_a(f_b(f_a(f_b(0))))$ , ...
- Interp. ist monoton:  $x > y \Rightarrow f_a(x) > f_a(y) \wedge f_b(x) > f_b(y)$
- Interp. ist kompatibel mit Programm:  $f_a(f_b(x)) > f_b(f_a(x))$
- resultierendes Constraint-System für  $P, Q, R, S$ ,
- Lösung mittels Z3

– Typeset by FoilTeX –

12

## Constraint-Programmierung—Beispiel

```
(set-logic QF_NIA)(set-option :produce-models true)
(declare-fun P () Int) (declare-fun Q () Int)
(declare-fun R () Int) (declare-fun S () Int)
(assert (and (< 0 P) (<= 0 Q) (< 0 R) (<= 0 S)))
(assert (> (+ (* P S) Q) (+ (* R Q) S)))
(check-sat) (get-value (P Q R S))
```

- *Constraint-System*: eine prädikatenlogische Formel  $F$   
 $0 < P \wedge \dots \wedge P \cdot S + Q > R \times Q + S$
- *Lösung*: Interpretation (Var.-Belegung), für die  $F$  wahr ist
- CP ist eine Form der *deklarativen* Programmierung.
- *Vorteil*: Benutzung von allgemeinen Suchverfahren (bereichs-, aber nicht anwendungsspezifisch).

– Typeset by FoilTeX –

13

## Constraints in der Unterhaltungsmathematik

- Nikoli (1980–, „the first puzzle magazine in Japan.“)  
<https://www.nikoli.co.jp/en/puzzles/>
- Erich Friedman: *Math Magic* 1998–2020  
<https://erich-friedman.github.io/mathmagic/>
- Simon Tatham's Portable Puzzle Collection <https://www.chiark.greenend.org.uk/~sgtatham/puzzles/>
- Angela und Otto Janko: <http://www.janko.at/Raetsel/>,
- Donald Knuth: *A Potpourri of Puzzles*, 2022,  
TAOCP, Band 4, Pre-Faszikel 9b,  
<https://cs.stanford.edu/~knuth/taocp.html>,  
<https://cs.stanford.edu/~knuth/fasc9b.ps.gz>

– Typeset by FoilTeX –

14

## Wettbewerbe für Constraint-Solver

- für aussagenlogische Formeln:  
<http://www.satcompetition.org/>  
(SAT = satisfiability)
- für prädikatenlogische Formeln  
<https://smt-comp.github.io/>  
(SMT = satisfiability modulo theories)  
Theorien:  $\mathbb{Z}$  mit  $\leq$ , Plus, Mal;  $\mathbb{R}$  mit  $\leq$ , Plus; ...
- Termination und Komplexität  
[https://www.termination-portal.org/wiki/Termination\\_Competition](https://www.termination-portal.org/wiki/Termination_Competition)

– Typeset by FoilTeX –

15

## Gliederung der Vorlesung

- Aussagenlogik
  - zur Modellierung (SAT-Kodierung) von Systemen mit endlichem Zustandsraum, begrenzter Schritt-Zahl (bounded model checking)
  - Werkzeuge: SAT-Kompiler (Tseitin-Transformation), SAT-Solver (Propagation, Backtracking, Lernen)
- eingeschränkte Prädikatenlogik
  - Unbekannte aus unendlichen Bereichen (Zahlen)
  - Werkzeuge: SMT-Solver (SAT modulo Theory)
- allgemeine Beweis-Systeme
  - zur Modellierung beliebiger (unbeschränkter) Systeme
  - Methode: Typisierung (Curry-Howard-Isomorphie,  $\text{program} = \text{proof}$ ), Werkzeuge: interaktive Beweiser

– Typeset by FoilTeX –

16

## Organisatorisches

- jede Woche 1 Vorlesung + 1 Übung
- Hausaufgaben (*Haus* bedeutet: zuhause bearbeiten, in der Übung diskutieren)
  - Aufgaben im Skript
  - Aufgaben in `autotool`Prüfungszulassung: regelmäßiges und erfolgreiches Bearbeiten von Hausaufgaben
- Klausur (2 h, keine Hilfsmittel).
- Quelltexte, Planung und Diskussion der Übungsaufgaben  
<https://gitlab.dit.hwk-leipzig.de/johannes.waldmann/fmw-ws24> (Projekt-Mitgliedschaft beantragen, Zugang wird dann auf Mitglieder eingeschränkt)

– Typeset by FoilTeX –

17

## Literatur

- Cerone, A., Roggenbach, M., Schlingloff, B.-H., Schneider, G., Shaikh, S.A.: *Teaching formal methods for software engineering - ten principles*, Informatica Didactica, Nr. 9, (2015). <https://www.informaticadidactica.de/uploads/Artikel/Schlinghoff2015/Schlinghoff2015.pdf>
- Uwe Schöning, Jacob Toran: *Das Erfüllbarkeitsproblem SAT*, Lehmanns (2012)
- Christel Baier, Joost Katoen: *Principles of Model Checking*. MIT Press, Cambridge (2008)
- Samuel Mimram: *Program = Proof*, <https://www.lix.polytechnique.fr/Labo/Samuel.Mimram/teaching/INF551/course.pdf>

– Typeset by FoilTeX –

18

## Hausaufgaben

WS 25: 1b, 5, 4a, 2.

1. Pierluigi Crescenzi, Viggo Kann *A compendium of NP optimization problems*  
<https://web.archive.org/web/20200227195925/http://www.nada.kth.se/~viggo/problemlist/compendium.html>

Beispiele:

- (a) Minimum File Transfer Scheduling (node195).
- (b) Minimum Dynamic Storage Allocation (node163).

Erläutern Sie die Spezifikation an einem Beispiel. Geben Sie eine lösbare Instanz sowie dafür zwei Lösungen mit unterschiedlichem Maß an.

– Typeset by FoilTeX –

19

2. Aufgabe: formalisieren Sie *Math Magic* Februar 2007.

Was ist dabei für Springer und König einfacher als für Dame, Läufer, Turm?

(allgemein für Math Magic: offene Fragen <https://erich-friedman.github.io/mathmagic/unsolved.html>)

3. Aufgabe: formalisieren Sie Wolkenkratzer oder Towers (was ist der Unterschied?)

<https://www.janko.at/Raetsel/Wolkenkratzer>  
<https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/towers.html>

- $B = \{0, 1, \dots, n-1\}$ , Unbekannte:  $h_{x,y} \in B$  für  $x, y \in B$
- Constraint für eine Zeile  $x$ : (entspr. für Spalte)

– Typeset by FoilTeX –

20

$\bigvee_{p \in \text{Permutationen}(0, \dots, n-1), p \text{ kompatibel mit Vorgaben}} \bigwedge_{y \in \{0, \dots, n-1\}} (h_{x,y} = p(y))$

Bsp:  $n = 4$ , Vorgabe links 2, rechts 1, kompatibel sind  $[0, 2, 1, 3], [2, 0, 1, 3], [2, 1, 0, 3], [2, 1, 0, 3]$ .

- diese Formel wird exponentiell groß (wg. Anzahl Permutationen),  
Folge-Aufgabe: *geht das auch polynomiell?*
- Geben Sie eine Aufgabenstellung der Größe  $w \times w$  an mit  $4 \cdot w$  Vorgaben (d.h., *alle* Vorgaben), die mehr als eine Lösung hat.  
... für  $w = 4$ , für größere  $w$  (einige, alle)
- (offene Frage?) Geben Sie eine eindeutig lösbare Instanz mit möglichst wenigen Vorgaben an. (Sind  $w - 1$  Vorgaben für  $w \times w$  immer möglich?)

– Typeset by FoilTeX –

21

4. Modellierung von Puzzles (aus Tatham-Collection)

(a) geben Sie ein Modell an für *Pegs* (Solitaire). Hinweise:

- Zustand = Boolesche Matrix,
- Schritt = Relation zwischen Matrizen,
- Lösung = Schrittfolge ( $\Rightarrow$  Zustandsfolge). Wieviele Schritte?

(b) Modell für *Unruly*. (keine Zustandsfolge, sondern direkt die Lösung. Welches sind die Unbekannten, welches sind ihre Beziehungen, untereinander und zur Vorgabe)

(c) modellieren Sie *Untangle*

Vergleichen Sie mit den tatsächlichen Quelltexten  
<https://git.tartarus.org/?p=simon/puzzles.git>

– Typeset by FoilTeX –

22

5. Constraint für monotone compatible Bewertungsfunktion:

- lösen Sie mit Z3 (ist im Pool installiert, vgl. <https://www.imn.hwk-leipzig.de/~waldmann/etc/pool/>)
- eine *kleinste* Lösung finden (Summe von  $P, Q, R, S$  möglichst klein) — dafür Assert(s) hinzufügen.
- Abstieg der so gefundenen Bewertungsfunktion nachrechnen für  $abab \rightarrow baab \rightarrow baba \rightarrow bbaa$
- gibt diese Bewertungsfunktion die maximale Schrittzahl genau wieder? (nein)
- Folge-Aufgabe: entspr. Constraint-System für Bewertungsfunktion für  $ab \rightarrow bba$  aufstellen und lösen.

– Typeset by FoilTeX –

23

# Erfüllbarkeit aussagenlogischer Formeln (SAT)

## Aussagenlogik: Syntax

aussagenlogische Formel:

- elementar: Variable  $v_1, \dots$
- zusammengesetzt: durch Operatoren
  - einstellig: Negation
  - zweistellig: Implikation, Äquivalenz, Antivalenz,
  - mehrstellig möglich: Konjunktion, Disjunktion,

damit auch Quantifikation über endlichen Bereich  $E$   
( $\forall x \in E : F$ ) ist (endliche!) Konjunktion  $\bigwedge_{x \in E} F$

## Aussagenlogik: Semantik

- Wertebereich  $\mathbb{B} = \{0, 1\}$ , Halbring  $(\mathbb{B}, \vee, \wedge, 0, 1)$

Übung: weitere Halbringe mit 2 Elementen?

- *Belegung* ist Abbildung  $b : V \rightarrow \mathbb{B}$
- *Wert* einer Formel  $F$  unter Belegung  $b$ :  $\text{val}(F, b)$
- wenn  $\text{val}(F, b) = 1$ , dann ist  $b$  ein *Modell* von  $F$ , Schreibweise:  $b \models F$
- *Modellmenge*  $\text{Mod}(F) = \{b \mid b \models F\}$
- $F$  *erfüllbar*, wenn  $\text{Mod}(F) \neq \emptyset$
- *Modellmenge einer Formelmenge*:  
 $\text{Mod}(M) = \{b \mid \forall F \in M : b \models F\}$

## Formulierung von SAT-Problemen mit Ersatz

- Autoren: Edward Kmett et al.,  
<https://hackage.haskell.org/package/ersatz>,
- ```
import Prelude hiding ((&&),(||),not )
import Ersatz
main = do
  ans <- solveWith minisat $ do
    p <- exists @Bit ; q <- exists @Bit
    assert $ (p || not q) && (not p || q)
    return [p,q]
  case ans of (Satisfied, Just res) -> print res
```
- Unbekannte erzeugen (`exists`), Formel konstruieren (`&&`,...), assertieren, lösen, Antwort benutzen
- zu Implementierung vgl. <https://www.imn.htwk-leipzig.de/~waldmann/etc/untutorial/ersatz/>

## Benutzung von SAT-Solvern

- Eingabeformat: SAT-Problem in CNF:
  - Variable = positive natürliche Zahl
  - Literal = ganze Zahl ( $\neq 0$ , mit Vorzeichen)
  - Klausel = Zeile, abgeschlossen durch 0.
  - Formel = Header `p cnf <#Variablen> <#Klauseln>`, danach Klauseln
- Beispiel: die (konj. Normal-)Formel  $(v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_2)$ :  

```
p cnf 2 2
1 -2 0
-1 2 0
```
- Löser (Bsp.): `minisat input.cnf output.text`  
Niklas Eén, Niklas Sörensson: *An Extensible SAT Solver*, 2003, <https://minisat.se/>,

## SAT-Modellierung für das $N$ -Damen-Problem

stelle möglichst viele Damen auf  $N \times N$ -Schachbrett, die sich nicht gegenseitig bedrohen.

- Unbekannte:  $q_{x,y}$  für  $(x,y) \in P = \{1, \dots, N\}^2$   
mit Bedeutung:  $q_{x,y} \iff$  Position  $(x,y)$  ist belegt
- Constraint:  $\bigwedge_{a,b \in F, a \text{ bedroht } b} \neg q_a \vee \neg q_b$ .
- „möglichst viele“ läßt sich hier vereinfachen zu:  
„in jeder Zeile genau eine“.  
vereinfachen zu: „... wenigstens eine.“

wir schreiben Programm,

## $N$ -Damen mit Ersatz

- Hilfsfunktionen für Boolesche Matrizen mit  

```
import qualified Ersatz.Relation as R
```
- ```
queens n = do
  out <- solveWith (cryptominisat5Path "kissat") $ do
    b <- R.relation ((1,1), (n,n))
    assert $ flip all (rows b) $ \ r -> or r
    assert $ flip all (R.indices b) $ \ p ->
      flip all (R.indices b) $ \ q ->
        encode (reaches p q) ==> not (b R.! p && b R.! q)
    return b
  case out of
    (Satisfied, Just b) -> do putStrLn $ R.table b
```
- (vollst. Quelltext: siehe Repo)

## Zusammenfassung Ersatz (bisher)

- innerhalb von `solveWith` steht Befehlsfolge, Befehl ändert Zustand (Variablenzähler, Klausel-Ausgabe)
  - `exists @Bit :: MonadSAT s m => m Bit, R.relation` – konstruiert neue Variable(n)
  - `assert :: Bit -> m ()` – übersetzt Formel in CNF, gibt Klauseln aus
- der Typ `Bit` beschreibt aussagenlogische Formeln (genauer: Schaltkreise), d.h., *symbolische* Repräsentation von (unbekannten) Wahrheitswerten  
Typ `Bool` beschreibt konkrete (bekannte) Wahrheitswerte  
booleschen Operatoren (`import Ersatz`) sind polymorph  
`not :: Boolean b => b -> b`,  
`instance Boolean Bool, instance Boolean Bit`

## Modellierung durch SAT: Ramsey

gesucht ist Kanten-2-Färbung des  $K_5$  ohne einfarbigen  $K_3$ .

- Aussagenvariablen  $f_{i,j}$  = Kante  $(i,j)$  ist rot (sonst blau).
- Constraints:  
$$\forall p : \forall q : \forall r : (p < q \wedge q < r) \Rightarrow ((f_{p,q} \vee f_{q,r} \vee f_{p,r}) \wedge \dots)$$

das ist ein Beispiel für ein Ramsey-Problem

(F. P. Ramsey, 1903–1930)

<http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Ramsey.html>

diese sind schwer, z. B. ist bis heute unbekannt: gibt es eine Kanten-2-Färbung des  $K_{43}$  ohne einfarbigen  $K_5$ ?

<http://www1.combinatorics.org/Surveys/ds1/sur.pdf>

## SAT-Modell für Peg-Solitaire

- Spielzug: Stein überspringen und entfernen,  
Aufgabe: existiert Zugfolge von Initial (gegeben)  
zu Final (genau ein Stein übrig)
- Beispie: Initial ist volles Rechteck minus ein Stein
- (Wdhlg.) aussagenlog. Modell als Folge von Relationen  $[B_0, B_1, \dots, B_n]$ , für die gilt  $\forall k : \text{step}(B_k, B_{k+1})$ .
- eine Realisierung:  $\text{move}(S, m, d, T)$  mit:  $S, T$  Brett,  $m$  Position (des übersprungenen Steins),  $d$  Richtung und  $\text{step}(S, T) = \bigvee_{m,d} \text{move}(S, m, d, T)$
- Finalität von  $B_n$  ist einfach: das Zählen muß nicht SAT-kodiert werden—warum?

– Typeset by FoilTeX –

32

## Hausaufgaben

WS 25: 2,3,4 (5, 7, 8 siehe Repo)

1. unterschiedliche Halbringe auf zwei Elementen?
2. für die Formel  $S(b, h)$  (abhängig von Parametern  $b, h \in \mathbb{N}$ )  
Variablen:  $v_{x,y}$  für  $1 \leq x \leq b, 1 \leq y \leq h$   
Constraints:
  - für jedes  $x$  gilt: wenigstens einer von  $v_{x,1}, v_{x,2}, \dots, v_{x,h}$  ist wahr
  - und für jedes  $y$  gilt: höchstens einer von  $v_{1,y}, v_{2,y}, \dots, v_{b,y}$  ist wahr(a) unter welcher Bedingung an  $b, h$  ist  $S(b, h)$  erfüllbar?  
Für den erfüllbaren Fall: geben Sie ein Modell an.

– Typeset by FoilTeX –

33

- Für den nicht erfüllbaren Fall: einen Beweis.
- (b) Erzeugen Sie (eine konjunktive Normalform für)  $S(b, h)$  durch ein Programm (Sprache/Bibliothek beliebig) ( $b, h$  von der Kommandozeile, Ausgabe nach stdout)
- (c) Lösen Sie  $S(b, h)$  durch minisat (kissat, Z3, ...), vergleichen Sie die Laufzeiten (auch im nicht erfüllbaren Fall).

3. Für  $S(b, h)$  (vorige Aufgabe): Formel-Konstruktion, Löser-Aufruf mit Ersatz.

- $v$  vom Typ `Relation Int Int` (vgl.  $N$  Damen)
- „für jedes  $x$ “: verwenden Sie `rows`
- „für jedes  $y$ “: schreiben und verwenden Sie entsprechende Fkt. `columns`
- „höchstens einer“: verwenden Sie „keine zwei“.

– Typeset by FoilTeX –

34

4. Für  $a, b \geq 2$ : die Ramsey-Zahl  $R(a, b)$  ist die kleinste Zahl  $n$ , für die gilt: jede rot-blau-Kantenfärbung eines  $K_n$  enthält einen roten  $K_a$  oder einen blauen  $K_b$ .  
(Der Satz von Ramsey ist, daß es für jedes  $a, b$  tatsächlich solche  $n$  gibt.)

- (a) Beweisen Sie:
- i.  $R(a, b) = R(b, a)$
  - ii.  $R(2, b) = b$
  - iii.  $R(a+1, b+1) \leq R(a, b+1) + R(a+1, b)$   
(das liefert einen Beweis des Satzes von Ramsey)
  - iv. wenn dabei beide Summanden rechts gerade Zahlen sind, dann  $R(a+1, b+1) < \dots$
- (b) Bestimmen Sie damit obere Schranken für  $R(3, 3)$ ,  $R(3, 4)$ ,  $R(4, 4)$  und vergleichen Sie mit den

– Typeset by FoilTeX –

35

unteren Schranken durch SAT-Kodierung.

5. SAT-Kodierung für  $R(a, b)$  mit Ersatz:

- `main = ramsey 3 3 5`

```
ramsey a b n = do
  out <- solveWith (cryptominisat5Path "kissat") $
    r <- R.symmetric_relation ((1,1), (n,n))
    assert $ flip all (subs a [1..n]) $ \ c ->
      flip any (subs 2 c) $ \ [x,y] ->
        r R.! (x,y)
    assert $ flip all (subs b [1..n]) $ \ c ->
      flip any (subs 2 c) $ \ [x,y] ->
        not $ r R.! (x,y)
  return r
...
```

- Hilfsfunktion: verteilten Teilfolgen gegebener Länge:

– Typeset by FoilTeX –

36

Beispiel: `subs 3 [1,2,3,4,5]`

`= [[1,2,3], [1,2,4], [1,2,5], ..., [3,4,5]]`  
(nicht notwendig in dieser Reihenfolge)

```
subs :: Int -> [a] -> [[a]]
subs 0 xs = [ [] ] ; subs k [] = []
subs k (x:xs) = map _ _ <> subs k xs
mit subs a [1..n] zur Auswahl des  $K_a$  sowie
subs 2 c zur Auswahl der Kanten.
```

- Diskutieren Sie Existenz (obere Schranke) und SAT-Kodierung für dreifarbigen Ramsey:  
 $R(a, b, c) :=$  das kleinste  $n$  mit: jeder Kanten-3-Färbung des  $K_n$  enthält einen roten  $K_a$  oder einen grünen  $K_b$  oder einen blauen  $K_c$ .  
Ergänzen und beweisen:  $R(a, b, c) \leq R(a, R(b, c))$ , anwenden für  $R(3, 3, 3)$ .

– Typeset by FoilTeX –

37

6. Modellierung als aussagenlogisches Constraint:

- Rösselsprung (= Hamiltonkreis)
- Norinori  
<https://nikoli.com/en/puzzles/norinori/>
- ABCEndView (oder ähnlich)  
<https://www.janko.at/Raetsel/AbcEndView/>

Vorgehen bei Modellierung:

- welches sind die Unbekannten, was ist deren Bedeutung?  
(Wie rekonstruiert man eine Lösung aus der Belegung, die der Solver liefert?)
- welches sind die Constraints?  
(wie stellt man sie in CNF dar? — falls nötig)

– Typeset by FoilTeX –

38

7. Unruly (S. Tatham Puzzles)

```
u1 = M.fromList -- eine Aufgabe (8 x 8)
$ map (,False) [(1,7), (3,2), (5,1), (5,3), (5,4), (5,
  <> map (,True) [(1,4), (6,2), (6,3), (7,5), (8,4), (8,
unruly u = do
  let bnd = ((1,1), (8,8))
  out <- solveWith (cryptominisat5Path "kissat") $
    b <- R.relation bnd
    assert $ flip all (M.toList u) $ \ (k,v) ->
      b R.! k == encode v
    assert $ flip all (rows b <> columns b) $ \ rc
      balanced rc && mixed 3 rc
  ...
```

wie kann man feststellen, daß es genau eine Lösung gibt? (Solver nochmals aufrufen für modifizierte Formel.)

– Typeset by FoilTeX –

39

### 8. Peg (S. Tatham Puzzles)

```
peg b = do
  let bnd = ((1,1), (b,b))
  full = A.genArray bnd $ \ i -> True
  start :: A.Array (Int,Int) Bool
  start = full A.// [((1,2), False)]
  out <- solveWith (cryptominisat5Path "kissat") $
    boards <- replicateM (b*b - 1) $ R.relation bnd
    assert $ R.equals (head boards) (encode start)
    assert $ all step $ zip boards $ drop 1 boards
  return boards
...
type Board = R.Relation Int Int

step :: (Board, Board) -> Bit
move :: Board -> (Int,Int) -> (Int,Int) -> Board ->
```

– Typeset by FoilTeX –

40

## Anwendg.: Bounded Model Checking

### Begriff, Motivation

- *model checking*: feststellen, ob
  - ein *Modell* eines realen Hard- oder Softwaresystems (z.B. Zustandsübergangssystem f. nebenläufiges Programm)
  - eine *Spezifikation* erfüllt (z.B. gegenseitiger Ausschluss, Liveness, Fairness)
- *symbolic model checking*:
  - symbolische Repräsentation von Zustandsfolgen
  - im Unterschied zu tatsächlicher Ausführung (Simulation)
- *bounded*: für Folgen beschränkter Länge

– Typeset by FoilTeX –

41

### Literatur, Software

- Armin Biere et al.: *Symbolic Model Checking without BDDs*, TACAS 1999, <http://fmv.jku.at/bmc/>  
Software damals: Übersetzung nach SAT, später: SMT (QB\_BV), Solver: <http://fmv.jku.at/boolector/>
- Daniel Kroening und Ofer Strichman: *Decision Procedures, an algorithmic point of view*, Springer, 2008.  
<http://www.decision-procedures.org/>  
Software: <http://www.cprover.org/cbmc/>
- Nikolaj Bjørner et al.: *Program Verification as Satisfiability Modulo Theories*, SMT-Workshop 2012, <http://smt2012.loria.fr/>  
Softw.: <https://github.com/Z3Prover/z3/wiki>

– Typeset by FoilTeX –

42

### BMC für Mutual Exclusion-Protokolle

System mit zwei (gleichartigen) Prozessen  $A, B$ :

```
A0: maybe goto A1
A1: if 1 goto A1 else goto A2
A2: 1 := 1; goto A3
A3: [critical;] goto A4
A4: 1 := 0; goto A0
```

```
B0: maybe goto B1
B1: if 1 goto B1 else goto B2
B2: 1 := 1; goto B3
B3: [critical;] goto B4
B4: 1 := 0; goto B0
```

Schließen sich A3 und B3 gegenseitig aus? (Nein.)

(nach: Donald E. Knuth: TAOCP, Vol. 4 Fasz. 6, S. 20ff)

– Typeset by FoilTeX –

43

### Modell: Zustandsübergangssystem

Zustände:

- jeder Zustand besteht aus:
  - Inhalte der Speicherstellen (hier:  $l \in \{0, 1\}$ )
  - Programmzähler (PC) jedes Prozesses (hier:  $A \in \{0 \dots 4\}, B \in \{0 \dots 4\}$ )
- Initialzustand:  $I = \{l = 0, A = 0, B = 0\}$
- Menge der Fehlerzustände:  $F = \{A = 3, B = 3\}$

Übergangsrelation (nichtdeterministisch): für  $P \in \{A, B\}$ :

- $P$  führt eine Aktion aus (schreibt Speicher, ändert PC)

Aussagenlog. Formel für  $I \rightarrow^{\leq k} F$  angeben, deren Erfüllbarkeit durch SAT- oder SMT-Solver bestimmen

– Typeset by FoilTeX –

44

### One-Hot-Kodierung

- $p \in \{0 \dots 4\}$  symbolisch repräsentieren durch Folge  $[p_0, \dots, p_4]$  von symbolischen Wahrheitswerten
  - ... von denen *genau einer* wahr ist
  - Bsp:  $p = 3$  kodiert durch  $[0, 0, 0, 1, 0]$ .
- das ist die *one hot*-Kodierung (eine Stelle ist *hot* = stromführend)
- eine Realisierung ist  $(\bigvee_i p_i) \wedge \bigwedge_{i < j} (\neg p_i \vee \neg p_j)$
- es gibt andere Kodierungen für endliche Bereiche (z.B.: binär: benötigt weniger Unbekannte, aber evtl. größere Formeln)

– Typeset by FoilTeX –

45

### Übung BMC

- Software: <https://git.imn.htwk-leipzig.de/waldmann/boumchak>
- überprüfe 1. gegenseitigen Ausschluss, 2. deadlock, 3. livelock (starvation) für weitere Systeme, z.B.  
E. W. Dijkstra, 1965:  
<https://www.cs.utexas.edu/~EWD/transcriptions/EWD01xx/EWD123.html#2.1>.  
G. L. Peterson, *Myths About the Mutual Exclusion Problem*, Information Processing Letters 12(3) 1981, 115–116

– Typeset by FoilTeX –

46

### Anzahl-Constraints

#### Definition, Motivation

- $\text{Count}_{\leq k}(x_1, \dots, x_n) := (\sum_i x_i) \leq k$ .
- AMO (at most one):  $= \text{Count}_{\leq 1}$ , entspr. ALO, EXO
- Schubfach-Constraint (als Testfall, erfüllbar gdw.  $B \leq H$ )  
 $\bigwedge_{1 \leq i \leq H} \text{AMO}(x_{ij} | 1 \leq j \leq B) \wedge \bigwedge_{1 \leq j \leq B} \text{ALO}(x_{ij} | 1 \leq i \leq H)$
- Schubfach für  $B = H$ : dann ist  $x$  Permutationsmatrix, repräsentiert Bijektion von  $\{1, \dots, B\}$  auf sich
- Anwend.: Rösselsprung, Hamiltonkreis in  $G = (V, E)$   
Pfad  $p$  in  $G$  als Bijektion von Indexmenge  $\{1, \dots, |V|\}$  in Knotenmenge  $V$  mit  $\bigwedge_i (p(i), p(i+1)) \in E$ .

– Typeset by FoilTeX –

47

## SAT-Kodierung eines Rösselsprungs

```
let n = height * width; places = [0 .. n-1]

let decode p = divMod p width
    edge p q =
        let (px,py) = decode p; (qx,qy) = decode q
        in 5 == (px-qx)^2 + (py-qy)^2
    rotate (x:xs) = xs <> [x]

a <- replicateM n $ replicateM n $ exists @Bit
assert $ all exactly_one a
assert $ all exactly_one $ transpose a
assert $ flip all (zip a $ rotate a) $ \ (e, f) ->
    flip all places $ \ p -> e!!p ==>
    flip any (filter (edge p) places) (\ q -> f!!q)
```

## SAT-Kodierungen von AMO (I) (quad, lin)

- quadratisch:  $\text{AMO}(x_1, \dots, x_n) = \bigwedge \{\bar{x}_i \vee \bar{x}_j \mid 1 \leq i < j \leq n\}$   
 $\binom{n}{2}$  Klauseln, keine zusätzlichen Variablen
- linear: mit Kodierung  $\text{enc} : x \mapsto (x_e, x_z) = (x \geq 1, x \geq 2)$ :  
 $0 \mapsto (0, 0), 1 \mapsto (1, 0), 2, 3, \dots \mapsto (1, 1)$   
 Addition:  $(x_e, x_z) +_{\text{enc}} (y_e, y_z) = \dots$   
 so daß  $\text{enc}(x + y) = \text{enc}(x) +_{\text{enc}} \text{enc}(y)$   
 $\text{AMO}(x_1, \dots, x_n) = \text{let } (s_e, s_z) = \sum_{\text{enc}} \text{enc}(x_i) \text{ in } \dots$

## SAT-Kodierungen von AMO (II) - log

- $\text{AMO}(x) = \exists h : (x_i \Rightarrow (i = h))$   
 $h$  binär repräsentiert mit  $\log n$  Bits.
- Bsp.  $\text{AMO}(x_0, \dots, x_3) = \exists h_1, h_0 : \begin{array}{l} (\bar{x}_0 \vee \bar{h}_1) \wedge (\bar{x}_0 \vee \bar{h}_0) \\ \wedge (\bar{x}_1 \vee \bar{h}_1) \wedge (\bar{x}_1 \vee h_0) \\ \wedge (\bar{x}_2 \vee h_1) \wedge (\bar{x}_2 \vee \bar{h}_0) \\ \wedge (\bar{x}_3 \vee h_1) \wedge (\bar{x}_3 \vee h_0) \end{array}$
- $n \log n$  Klauseln,  $\log n$  zusätzliche Variablen
- die Hilfsvariablen  $h_0, h_1$  sind keine Funktionen der Eingangsvariablen. (wenn alle  $x_i$  falsch, dann  $h_i$  beliebig)
- Ü: man kann eine Skolem-Funktion trotzdem einfach angeben

## SAT-Kodierungen von AMO (III) - sqrt

- für  $\text{AMO}(x)$ : die  $x$  in einem Rechteck anordnen,  
 $z_i := \bigvee_j x_{ij}$  (Zeile  $i$ ),  $s_j := \bigvee_i x_{ij}$  (Spalte  $j$ ),  
 dann  $\text{AMO}(x) = \text{AMO}(z) \wedge \text{AMO}(s)$ .
- Jingchao Chen: *A New SAT Encoding of the At-Most-One Constraint* 10th Workshop Constraint Modeling and Reformulation, 2010 <https://www.it.uu.se/research/group/astra/ModRef10/programme.html>
- Formelgröße  $f(n) = \Theta(n) + 2f(\sqrt{n})$ , mit  $\Theta(\sqrt{n})$  Hilfsvar.  
 Lineare Faktoren sind klein. Ü: wenn `assert (amo xs)`, kann man einige Klauseln weglassen. Welche?

## Aufgaben

WS 25: 2, 7, 11

1. Modellierung Sudoku: Kodierung eines (ausgefüllten) Schemas durch  $9 \times 9 \times 9$  unbekannte Bit mit  $u_{x,y,z} \iff$  auf Koordinate  $(x, y)$  steht Zahl  $z$ .  
 für welche Teilmengen gelten EXO-Constraints?  
 Hinweis: für  $9^2 + 3 \cdot 9^2$  Mengen, jede mit 9 Elementen.
2. geben Sie Beispiele aus Rätsel-Sammlungen von Nikoli, Janko, Tatham, bei deren SAT-Kodierung AMO- oder EXO-Constraints vorkommen sollten
3. Vergleichen Sie die *commander*-Kodierung für AMO von Klieber und Kwon, Int. Workshop Constraints in Formal Verification, 2007, mit Kodierungen aus dem Skript.

- a) auf dem Papier, b) praktisch: mit ersatz implementieren, Formelgrößen messen, auch nach Vorverarbeitung durch minisat
4. für AMO-log: geben Sie eine Skolem-Funktion für  $\forall x_0, \dots \exists h_0, \dots : \dots$  an.  
 d.h., die eine erfüllende Belegung der  $h_i$  bestimmt, falls  $\text{AMO}(x_0, \dots)$  wahr ist.
5. für AMO über  $2^k$  Argumente: verwenden Sie sqrt-Kodierung für Rechteck mit Abmessungen  $2 \times 2^{k-1}$ , dann rekursiv über  $2^{k-1}$ .  
 Vergleichen Sie mit der log-Kodierung.
6. für AMO-lin: untersuchen Sie den Unterschied zwischen der Verwendung von `foldr` (von rechts) und `foldb` (balanciert)

welche Maße der erzeugten Formel stimmen überein, welche unterscheiden sich?

7. vergleichen Sie Formelgrößen und Solver-Laufzeiten für unterschiedliche AMO-Kodierungen für die Spalten in den unlösbaren Schubfach-Formeln  $S(n+1, n)$ .
8. für die AMO-Kodierungen linear und sqrt: wie kann man mit möglichst wenig Zusatz-Aufwand EXO erhalten?
9. ordnen Sie die EXO-Kodierung im Bounded Model Checker (boumchak) in die Systematik der VL ein.
10. Für den Rösselsprung (Code in `fmw-24/leap`)
  - gegebene EXO-Implementierung diskutieren, durch andere ersetzen, Formelgröße/Solverlaufzeit beobachten
  - zusätzliches `assert $ a !! 0 !! 0` diskutieren

- andere Sprung-Distanzen (Giraffe usw., siehe unten)
- Kamel (1,3)-Sprung (auf nur schwarzen? oder weißen?) Feldern implementieren
- andere Kodierung für Hamiltonkreis: Neng-Fa Zhou: *In Pursuit of an Efficient SAT Encoding for the Hamiltonian Cycle Problem*, CP 2020, ModRef 2019, <https://www.sci.brooklyn.cuny.edu/~zhou/>
- Anwendung: George Jelliss: *Leapers at Large*, 2001, 2022 <http://www.mayhematics.com/t/pl.htm>  
 Resultate zur Existenz von Hamilton-Pfaden (HP) und -Kreisen (HC) nachrechnen und ergänzen, Bsp.:
  - HP für Giraffe (1,4) auf  $11 \times 11$ ?  
 (Nein—unsat nach 6 Stunden)
  - HP für Zebra (2,3) auf  $13 \times 13$ ?

– HP für Antilope (3, 4) auf  $20 \times 20$ ?

vgl. Donald Knuth: *Leaper Graphs*, 1994,

<https://arxiv.org/abs/math/9411240>

1. Plan SAT-Kodierung Skyline (Towers, Wolkenkratzer)

Jede Zahl (jedes Haus) wird one-hot-kodiert, dann

```
type Haus=[Bit];type Zeile=[Haus];type Stadt=[Zeile
```

- (ohne Vorgaben) welche AMO-Constraints gelten?
- (mit Vorgaben) Implementieren Sie die Berechnung der (von links) sichtbaren Dächer einer Zeile  
`sicht :: Zeile -> [Bit]`. Das Resultat soll der Bitvektor  $s$  sein mit  $s_i \iff$  Dach der Höhe  $i$  ist sichtbar.
- für die anderen Blickrichtungen: die Funktion `sicht` bleibt, die `Stadt` wird transformiert. Wie?