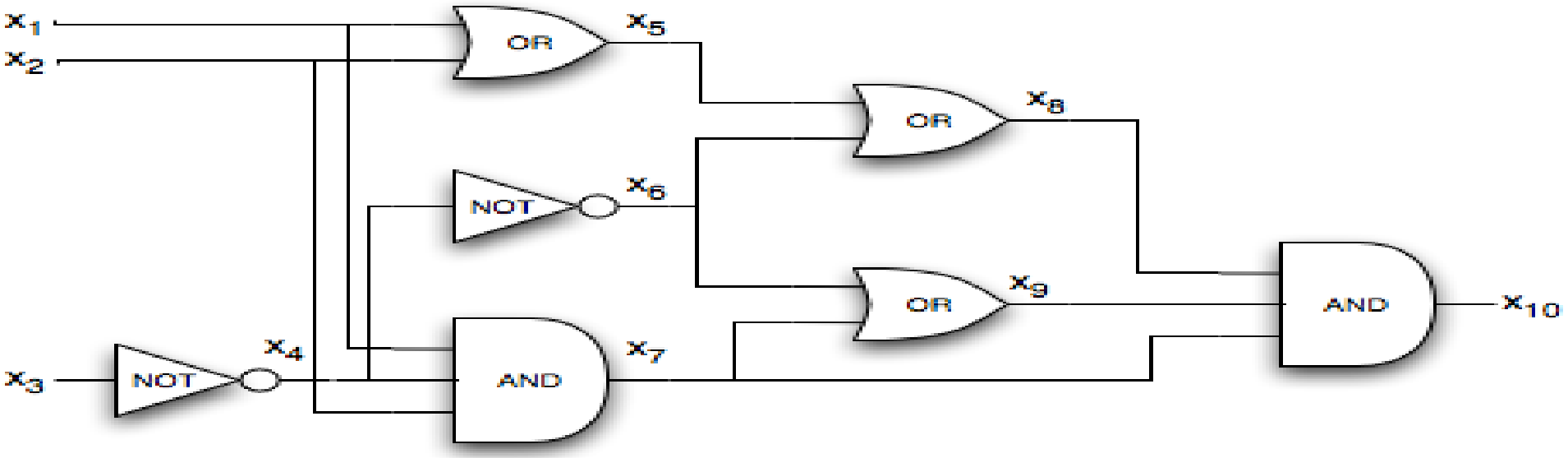
Karlsruhe Institute of Technology

Practical SAT Solving

Lecture 11

Carsten Sinz, Tomáš Balyo | July 24, 2017

INSTITUTE FOR THEORETICAL COMPUTER SCIENCE



Lecture Outline

- Introduction: Satisfiability Modulo Theory (SMT)
- DPLL(T) (Slides from Stephan Falke)
- QF_BV (Slides from Stephan Falke)

SMT: Motivation

- Propositional logic quite “low-level” for encoding many practical problems
- Often arithmetic involved (e.g. in hardware / program verification)
- Examples:
  - Linear (integer/real) arithmetic:

$$x + y < 5 \wedge (2x - y > 4 \vee x + y > 7)$$

  - Non-linear arithmetic:

$$x^2 + y^2 = 4 \wedge x - y = 3$$

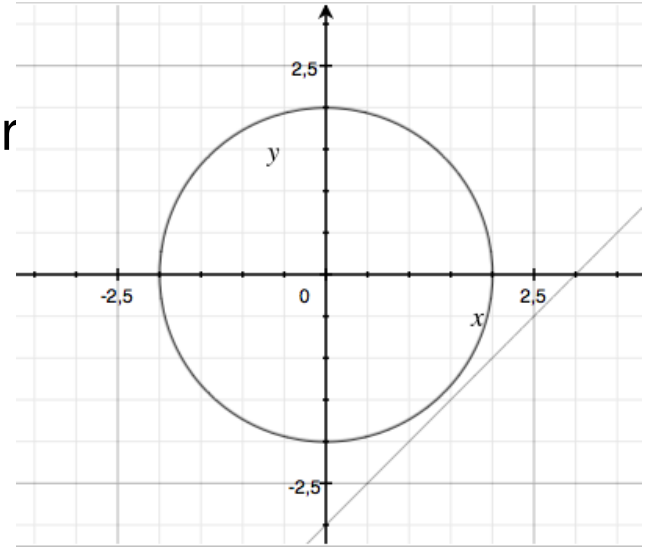
  - “Program arithmetic”:

$$x^2 \equiv 31 \pmod{2^{32}}$$

SMT: Motivation

- Propositional logic quite “low-level” for encoding many practical problems
- Often arithmetic involved (e.g. in hardwar
- Examples:
  - Linear (integer/real) arithmetic:

$$x + y < 5 \wedge (2x - y >$$



  - Non-linear arithmetic:

$$x^2 + y^2 = 4 \wedge x - y = 3$$

  - “Program arithmetic”:

$$x^2 \equiv 31 \pmod{2^{32}}$$

What is SMT?

- Extend propositional logic by certain theories of first-order logic.
- Theories are mainly about arithmetic (linear integer arithmetic, non-linear real arithmetic, etc.)
- In most cases quantifier-free fragment of first-order logic (because of decidability).
  - Note: The first-order theory of reals with $+$, $*$, $=$ and $<$ is decidable (Tarski).
- Syntax and semantics of SMT standardized in SMT-LIB standard.

SMT-LIB

SMT-LIB is an international initiative aimed at facilitating research and development in Satisfiability Modulo Theories (SMT). Since its inception in 2003, the initiative has pursued these aims by focusing on the following concrete goals.

- Provide standard rigorous descriptions of background theories used in SMT systems.
- Develop and promote common input and output languages for SMT solvers.
- Connect developers, researchers and users of SMT, and develop a community around it.
- Establish and make available to the research community a large library of benchmarks for SMT solvers.
- Collect and promote software tools useful to the SMT community.

This website provides access to the following main artifacts of the initiative.

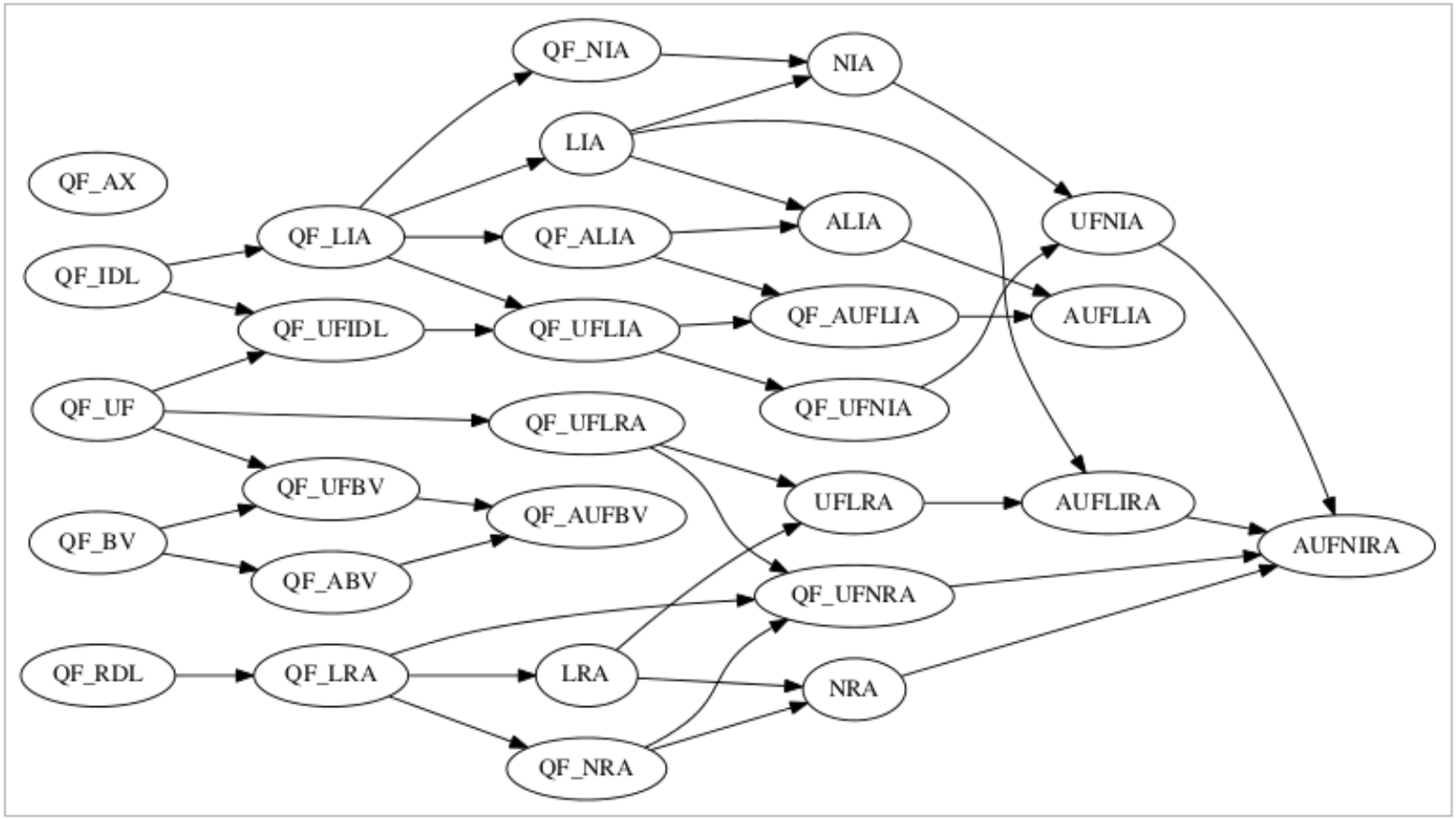
- Documents describing the SMT-LIB input/output language for SMT solvers and its semantics;
- Specifications of background theories and *logics*;
- A large library of input problems, or benchmarks, written in the SMT-LIB language.
- Links to SMT solvers and related tools and utilities.

Home
About
News
Standard
- Language
- Theories
- Logics
- Examples

Benchmarks
Software
- Solvers
- Utilities

Contact
Related
Credits

SMT-LIB: Logics



Theorien 1

Definition

Eine Theorie T besteht aus

- einer Signatur Σ bestehend aus Konstanten-, Funktions-, und Prädikatensymbolen
- einer Menge Ax von Axiomen (Sätze der Prädikatenlogik)

- Die Symbole in Σ haben keine festgelegte Bedeutung
- Die Bedeutung wird erst durch Ax fest gelegt

Theorien 1

Definition

Eine Theorie T besteht aus

- einer Signatur Σ bestehend aus Konstanten-, Funktions-, und Prädikatensymbolen
- einer Menge Ax von Axiomen (Sätze der Prädikatenlogik)

- Die Symbole in Σ haben keine festgelegte Bedeutung
- Die Bedeutung wird erst durch Ax fest gelegt

Theorien 1

Definition

Eine Theorie T besteht aus

- einer Signatur Σ bestehend aus Konstanten-, Funktions-, und Prädikatensymbolen
- einer Menge Ax von Axiomen (Sätze der Prädikatenlogik)

- Die Symbole in Σ haben keine festgelegte Bedeutung
- Die Bedeutung wird erst durch Ax fest gelegt

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome *Ax*:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Theorie der Uninterpretierten Funktionen

Beispiel

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. für jedes n-stellige Funktionssymbol $f \in \Sigma$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema das unendliche viele Axiome repräsentiert

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\, \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\, x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\, F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\, F[x]$ (Induktion)
  5. $\forall x.\, x + 0 = x$ (Plus Null)
  6. $\forall x, y.\, x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\, x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\, x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 1

Karlsruhe Institute of Technology

Beispiel

(Peano Arithmetik)

- $\Sigma = \{=, 0, 1, +, *\}$
- Axiome Ax:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
  7. $\forall x.\ x * 0 = 0$ (Multiplikation Null)
  8. $\forall x, y.\ x * (y + 1) = (x * y) + x$ (Multiplikation Nachfolger)
- (Induktion) ist ein Axiomschema

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel
(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel
(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel

(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel

(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Arithmetische Theorien 2

Beispiel

(Presburger Arithmetik)

- $\Sigma = \{=, 0, 1, +\}$ keine Multiplikation!
- Axiome *Ax*:
  1. Axiome der Theorie der Uninterpretierten Funktionen
  2. $\forall x.\ \neg(x + 1 = 0)$ (Null)
  3. $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$ (Nachfolger)
  4. $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$ (Induktion)
  5. $\forall x.\ x + 0 = x$ (Plus Null)
  6. $\forall x, y.\ x + (y + 1) = (x + y) + 1$ (Plus Nachfolger)
- (Induktion) ist ein Axiomschema

Beispiel

(Lineare Arithmetik ganzer Zahlen)

- $\Sigma = \{=, >, +, -, \ldots, -2, -1, 0, 1, 2, \ldots\}$ keine Multiplikation!
- Axiome *Ax*: ...

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome *Ax*:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorie der Arrays

Beispiel

- $\Sigma = \{=, \text{read}, \text{write}\}$
- Axiome Ax:
  1. $\forall x.\ x = x$ (Reflexivität)
  2. $\forall x, y.\ x = y \rightarrow y = x$ (Symmetrie)
  3. $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$ (Transitivität)
  4. $\forall a, i, j.\ i = j \rightarrow \text{read}(a, i) = \text{read}(a, j)$ (Array Kongruenz)
  5. $\forall a, v, i, j.\ i = j \rightarrow \text{read}(\text{write}(a, i, v), j) = v$ (read-über-write 1)
  6. $\forall a, v, i, j.\ i \neq j \rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$ (read-über-write 2)
  7. $\forall a, b.\ a = b \leftrightarrow (\forall i.\ \text{read}(a, i) = \text{read}(b, i))$ (Extensionalität)

Theorien 2

Definition

Eine Σ-Formel F ist T-erfüllbar gdw. $Ax \cup \{F\}$ erfüllbar ist

Definition

Eine Σ-Formel F ist T-gültig gdw. $Ax \models F$

Theorien 2

Definition

Eine Σ-Formel F ist T-erfüllbar gdw. $Ax \cup \{F\}$ erfüllbar ist

Definition

Eine Σ-Formel F ist T-gültig gdw. $Ax \models F$

Theorien 3

Definition

Eine Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel entschieden werden kann

Definition

Ein Fragment einer Theorie T ist eine syntaktisch eingeschränkte Teilmenge aller Σ-Formeln

- Beispiel: Quantoren-freies Fragment (QFF)

Definition

Eine Fragment einer Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel des Fragments entschieden werden kann

Theorien 3

Definition

Eine Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel entschieden werden kann

Definition

Ein Fragment einer Theorie T ist eine syntaktisch eingeschränkte Teilmenge aller Σ-Formeln

- Beispiel: Quantoren-freies Fragment (QFF)

Definition

Eine Fragment einer Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel des Fragments entschieden werden kann

Theorien 3

Definition

Eine Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel entschieden werden kann

Definition

Ein Fragment einer Theorie T ist eine syntaktisch eingeschränkte Teilmenge aller Σ-Formeln

- Beispiel: Quantoren-freies Fragment (QFF)

Definition

Eine Fragment einer Theorie T ist entscheidbar gdw. T-Erfüllbarkeit für jede Σ-Formel des Fragments entschieden werden kann

Entscheidbarkeit

Theorie	Entscheidbar?	QFF entscheidbar?
Uninterpretierte Funktionen	—	√
Peano Arithmetik	—	—
Presburger Arithmetik	√	√
Lineare Arithmetik ganzer Zahlen	√	√
Arrays	—	√

DPLL(T): Idee

- Im folgenden: nur quantoren-freie Σ-Formeln
- Benutze DPLL um Konjunktionen von T-Literalen zu erzeugen
  - diese T-Literale erfüllen das Boolesche Skelett der Formel
- Prüfe ob die Konjunktion konsistent in T ist
  - Falls ja, extrahiere ein T-Modell
  - Falls nein, schließe diesen "Kandidaten" aus

DPLL(T): Idee

- Im folgenden: nur quantoren-freie Σ-Formeln
- Benutze DPLL um Konjunktionen von T-Literalen zu erzeugen
  - diese T-Literale erfüllen das Boolesche Skelett der Formel
- Prüfe ob die Konjunktion konsistent in T ist
  - Falls ja, extrahiere ein T-Modell
  - Falls nein, schließe diesen "Kandidaten" aus

DPLL(T): Idee

- Im folgenden: nur quantoren-freie Σ-Formeln
- Benutze DPLL um Konjunktionen von T-Literalen zu erzeugen
  - diese T-Literale erfüllen das Boolesche Skelett der Formel
- Prüfe ob die Konjunktion konsistent in T ist
  - Falls ja, extrahiere ein T-Modell
  - Falls nein, schließe diesen "Kandidaten" aus

DPLL(T): Idee

- Im folgenden: nur quantoren-freie Σ-Formeln
- Benutze DPLL um Konjunktionen von T-Literalen zu erzeugen
  - diese T-Literale erfüllen das Boolesche Skelett der Formel
- Prüfe ob die Konjunktion konsistent in T ist
  - Falls ja, extrahiere ein T-Modell
  - Falls nein, schließe diesen "Kandidaten" aus

DPLL(T): Idee

- Im folgenden: nur quantoren-freie Σ-Formeln
- Benutze DPLL um Konjunktionen von T-Literalen zu erzeugen
  - diese T-Literale erfüllen das Boolesche Skelett der Formel
- Prüfe ob die Konjunktion konsistent in T ist
  - Falls ja, extrahiere ein T-Modell
  - Falls nein, schließe diesen "Kandidaten" aus

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0)$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E)$$

Modell für das Boolesche Skelett:

$$A,\ C,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ y < 1,\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge y < 1 \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 1)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \ \wedge \ [\neg(y \geq 1) \vee \neg(y < 1)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \ \wedge \ (\neg A \vee \neg C)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ D$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ x \geq 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge x \geq 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(x < 0) \vee \neg(x \geq 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \wedge [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \ \wedge \ [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \ \wedge \ (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \ \wedge \ [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \ \wedge \ (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \ \wedge \ [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \ \wedge \ (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \ \wedge \ [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \ \wedge \ (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \ \wedge \ [\neg(x < 0) \vee \neg(x \geq 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \ \wedge \ (\neg B \vee \neg D)$$

Modell für das Boolesche Skelett:

$$A,\ \neg C,\ B,\ \neg D,\ E$$

Entsprechende T-Literale:

$$y \geq 1,\ \neg(y < 1),\ x < 0,\ \neg(x \geq 0),\ y < 0$$

Inkonsistente Konjunktion von T-Literalen:

$$y \geq 1 \wedge \neg(y < 1) \wedge x < 0 \wedge \neg(x \geq 0) \wedge y < 0$$

Schließe diesen “Kandidaten” aus:

$$\neg(y \geq 1) \vee \neg(y < 0)$$

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \wedge [\neg(x < 0) \vee \neg(x \geq 0)] \wedge [\neg(y \geq 1) \vee \neg(y < 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg D) \wedge (\neg A \vee \neg E)$$

Boolesches Skelett hat kein Modell

$\Longrightarrow$ Ursprüngliche Formel hat kein T-Modell

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \wedge [\neg(x < 0) \vee \neg(x \geq 0)] \wedge [\neg(y \geq 1) \vee \neg(y < 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg D) \wedge (\neg A \vee \neg E)$$

Boolesches Skelett hat kein Modell

$\Longrightarrow$ Ursprüngliche Formel hat kein T-Modell

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \wedge [\neg(x < 0) \vee \neg(x \geq 0)] \wedge [\neg(y \geq 1) \vee \neg(y < 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg D) \wedge (\neg A \vee \neg E)$$

Boolesches Skelett hat kein Modell

$\Longrightarrow$ Ursprüngliche Formel hat kein T-Modell

Beispiel

Lineare Arithmetik ganzer Zahlen:

$$y \geq 1 \wedge (x < 0 \vee y < 1) \wedge (x \geq 0 \vee y < 0) \wedge [\neg(y \geq 1) \vee \neg(y < 1)] \wedge [\neg(x < 0) \vee \neg(x \geq 0)] \wedge [\neg(y \geq 1) \vee \neg(y < 0)]$$

Boolesches Skelett:

$$A \wedge (B \vee C) \wedge (D \vee E) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg D) \wedge (\neg A \vee \neg E)$$

Boolesches Skelett hat kein Modell

$\Longrightarrow$ Ursprüngliche Formel hat kein T-Modell

Inhalt

Inhalt

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------
  00101100   44
```

Überlauf!

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------------
  00101100   44
```

Überlauf!

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------
  00101100   44
```

Überlauf!

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------------
  00101100   44
```

Überlauf!

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------------
  00101100   44
```

Überlauf!

Beispiel

Was gibt das folgende Programm aus?

```
...
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
...
```

300? 44? 42?

Falls `unsigned char` 8 bit belegt:

```
  11001000  200
+ 01100100  100
------------------
  00101100   44
```

Überlauf!

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $x - y > 0 \wedge \neg(x > y)$ LIA-erfüllbar? NEIN

Aber: Assertion schlägt fehl, falls $\mathrm{x} = -2147483648$ und $\mathrm{y} = 1$

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $x - y > 0 \wedge \neg(x > y)$ LIA-erfüllbar? NEIN

Aber: Assertion schlägt fehl, falls $x = -2147483648$ und $y = 1$

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $x - y > 0 \wedge \neg(x > y)$ LIA-erfüllbar? NEIN

Aber: Assertion schlägt fehl, falls $x = -2147483648$ und $y = 1$

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $x - y > 0 \wedge \neg(x > y)$ LIA-erfüllbar? NEIN

Aber: Assertion schlägt fehl, falls $\mathtt{x} = -2147483648$ und $\mathtt{y} = 1$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathrm{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  ...
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  ...

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Bitvektorlogik

Theorie

- Funktionssymbole:
  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n-1}$
  $\mathtt{concat}, (\mathtt{extract}_{i,j})_{0 \leq j < i}$
  $(\mathtt{zero\_extend}_n)_{n>0}, (\mathtt{sign\_extend}_n)_{n>0}$
  $\mathtt{bvnot}, \mathtt{bvand}, \mathtt{bvor}$
  $\mathtt{bvshl}, \mathtt{bvlshr}, \mathtt{bvashr}$
  $\mathtt{bvadd}, \mathtt{bvsub}, \mathtt{bvmul}$
  $\ldots$
- Prädikatensymbole:
  $=, \mathtt{bvult}, \mathtt{bvule}, \mathtt{bvslt}, \mathtt{bvsle}$
- Axiome:
  $\ldots$

Werte

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor. Die Breite von b ist $|b| = n$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenlose Wert von b ist

$$\langle b \rangle_U = \sum_{i=0}^{n-1} b_i * 2^i$$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenbehaftete Wert (Zweierkomplement) von b ist

$$\langle b \rangle_S = -2^{n-1} * b_{n-1} + \sum_{i=0}^{n-2} b_i * 2^i$$

Werte

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor. Die Breite von b ist $|b| = n$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenlose Wert von b ist

$$\langle b \rangle_U = \sum_{i=0}^{n-1} b_i * 2^i$$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenbehaftete Wert (Zweierkomplement) von b ist

$$\langle b \rangle_S = -2^{n-1} * b_{n-1} + \sum_{i=0}^{n-2} b_i * 2^i$$

Werte

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor. Die Breite von b ist $|b| = n$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenlose Wert von b ist

$$\langle b \rangle_U = \sum_{i=0}^{n-1} b_i * 2^i$$

Definition

Sei $b = (b_{n-1} \cdots b_0)$ ein Bitvektor der Breite $n > 0$. Der vorzeichenbehaftete Wert (Zweierkomplement) von b ist

$$\langle b \rangle_S = -2^{n-1} * b_{n-1} + \sum_{i=0}^{n-2} b_i * 2^i$$

Semantik 1

- $|\mathrm{bv}_{n,k}| = n$

  $\langle \mathrm{bv}_{n,k} \rangle_U = k$

- Intuition:

  $\mathtt{concat}\ (x_{n-1} \cdots x_0)\ (y_{m-1} \cdots y_0) = (x_{n-1} \cdots x_0 y_{m-1} \cdots y_0)$

  $|\mathtt{concat}\ x\ y| = |x| + |y|$

  $$(\mathtt{concat}\ x\ y)_k = \begin{cases} y_k & 0 \leq k < |y| \\ x_{k-|y|} & |y| \leq k < |x| + |y| \end{cases}$$

- Intuition: $\mathtt{extract}_{i,j}\ (x_{n-1} \cdots x_0) = (x_i \cdots x_j)$

  $\mathtt{extract}_{i,j}\ x$ ist nur definiert falls $i < |x|$

  $|\mathtt{extract}_{i,j}\ x| = i - j + 1$

  $(\mathtt{extract}_{i,j}\ x)_k = x_{j+k}$

Semantik 1

- $|\mathrm{bv}_{n,k}| = n$

  $\langle \mathrm{bv}_{n,k} \rangle_U = k$

- Intuition:
  $\mathtt{concat}\ (x_{n-1} \cdots x_0)\ (y_{m-1} \cdots y_0) = (x_{n-1} \cdots x_0 y_{m-1} \cdots y_0)$

  $|\mathtt{concat}\ x\ y| = |x| + |y|$

  $$(\mathtt{concat}\ x\ y)_k = \begin{cases} y_k & 0 \leq k < |y| \\ x_{k-|y|} & |y| \leq k < |x| + |y| \end{cases}$$

- Intuition: $\mathtt{extract}_{i,j}\ (x_{n-1} \cdots x_0) = (x_i \cdots x_j)$

  $\mathtt{extract}_{i,j}\ x$ ist nur definiert falls $i < |x|$

  $|\mathtt{extract}_{i,j}\ x| = i - j + 1$

  $(\mathtt{extract}_{i,j}\ x)_k = x_{j+k}$

Semantik 1

- $|\mathrm{bv}_{n,k}| = n$

  $\langle \mathrm{bv}_{n,k} \rangle_U = k$

- Intuition:

  $\mathtt{concat}\ (x_{n-1} \cdots x_0)\ (y_{m-1} \cdots y_0) = (x_{n-1} \cdots x_0 y_{m-1} \cdots y_0)$

  $|\mathtt{concat}\ x\ y| = |x| + |y|$

  $$(\mathtt{concat}\ x\ y)_k = \begin{cases} y_k & 0 \leq k < |y| \\ x_{k-|y|} & |y| \leq k < |x| + |y| \end{cases}$$

- Intuition: $\mathtt{extract}_{i,j}\ (x_{n-1} \cdots x_0) = (x_i \cdots x_j)$

  $\mathtt{extract}_{i,j}\ x$ ist nur definiert falls $i < |x|$

  $|\mathtt{extract}_{i,j}\ x| = i - j + 1$

  $(\mathtt{extract}_{i,j}\ x)_k = x_{j+k}$

Semantik 1

- $|\mathrm{bv}_{n,k}| = n$

  $\langle \mathrm{bv}_{n,k} \rangle_U = k$

- Intuition:

  $\mathtt{concat}\ (x_{n-1} \cdots x_0)\ (y_{m-1} \cdots y_0) = (x_{n-1} \cdots x_0 y_{m-1} \cdots y_0)$

  $|\mathtt{concat}\ x\ y| = |x| + |y|$

  $$(\mathtt{concat}\ x\ y)_k = \begin{cases} y_k & 0 \leq k < |y| \\ x_{k-|y|} & |y| \leq k < |x| + |y| \end{cases}$$

- Intuition: $\mathtt{extract}_{i,j}\ (x_{n-1} \cdots x_0) = (x_i \cdots x_j)$

  $\mathtt{extract}_{i,j}\ x$ ist nur definiert falls $i < |x|$

  $|\mathtt{extract}_{i,j}\ x| = i - j + 1$

  $(\mathtt{extract}_{i,j}\ x)_k = x_{j+k}$

Semantik 1

- $|\mathrm{bv}_{n,k}| = n$

  $\langle \mathrm{bv}_{n,k} \rangle_U = k$

- Intuition:

  $\mathtt{concat}\ (x_{n-1} \cdots x_0)\ (y_{m-1} \cdots y_0) = (x_{n-1} \cdots x_0 y_{m-1} \cdots y_0)$

  $|\mathtt{concat}\ x\ y| = |x| + |y|$

  $$(\mathtt{concat}\ x\ y)_k = \begin{cases} y_k & 0 \leq k < |y| \\ x_{k-|y|} & |y| \leq k < |x| + |y| \end{cases}$$

- Intuition: $\mathtt{extract}_{i,j}\ (x_{n-1} \cdots x_0) = (x_i \cdots x_j)$

  $\mathtt{extract}_{i,j}\ x$ ist nur definiert falls $i < |x|$

  $|\mathtt{extract}_{i,j}\ x| = i - j + 1$

  $(\mathtt{extract}_{i,j}\ x)_k = x_{j+k}$

Semantik 2

- Intuition: $\texttt{zero\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{0 \cdots 0}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

$$|\texttt{zero\_extend}_n\ x| = |x| + n$$

$$(\texttt{zero\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ 0 & k \geq |x| \end{cases}$$

- Intuition: $\texttt{sign\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{x_{m-1} \cdots x_{m-1}}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

$$|\texttt{sign\_extend}_n\ x| = |x| + n$$

$$(\texttt{sign\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ x_{|x|-1} & k \geq |x| \end{cases}$$

- $\langle b \rangle_U = \langle \texttt{zero\_extend}_n\ b \rangle_U$ $\qquad$ $\langle b \rangle_S = \langle \texttt{sign\_extend}_n\ b \rangle_S$

Semantik 2

- Intuition: $\texttt{zero\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{0 \cdots 0}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{zero\_extend}_n\ x| = |x| + n$

  $(\texttt{zero\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ 0 & k \geq |x| \end{cases}$

- Intuition: $\texttt{sign\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{x_{m-1} \cdots x_{m-1}}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{sign\_extend}_n\ x| = |x| + n$

  $(\texttt{sign\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ x_{|x|-1} & k \geq |x| \end{cases}$

- $\langle b \rangle_U = \langle \texttt{zero\_extend}_n\ b \rangle_U$ $\qquad$ $\langle b \rangle_S = \langle \texttt{sign\_extend}_n\ b \rangle_S$

Semantik 2

- Intuition: $\texttt{zero\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{0 \cdots 0}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{zero\_extend}_n\ x| = |x| + n$

  $(\texttt{zero\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ 0 & k \geq |x| \end{cases}$

- Intuition: $\texttt{sign\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{x_{m-1} \cdots x_{m-1}}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{sign\_extend}_n\ x| = |x| + n$

  $(\texttt{sign\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ x_{|x|-1} & k \geq |x| \end{cases}$

- $\langle b \rangle_U = \langle \texttt{zero\_extend}_n\ b \rangle_U$ $\qquad$ $\langle b \rangle_S = \langle \texttt{sign\_extend}_n\ b \rangle_S$

Semantik 2

- Intuition: $\texttt{zero\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{0 \cdots 0}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{zero\_extend}_n\ x| = |x| + n$

  $$(\texttt{zero\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ 0 & k \geq |x| \end{cases}$$

- Intuition: $\texttt{sign\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{x_{m-1} \cdots x_{m-1}}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{sign\_extend}_n\ x| = |x| + n$

  $$(\texttt{sign\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ x_{|x|-1} & k \geq |x| \end{cases}$$

- $\langle b \rangle_U = \langle \texttt{zero\_extend}_n\ b \rangle_U$ $\quad$ $\langle b \rangle_S = \langle \texttt{sign\_extend}_n\ b \rangle_S$

Semantik 2

- Intuition: $\texttt{zero\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{0 \cdots 0}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{zero\_extend}_n\ x| = |x| + n$

  $$(\texttt{zero\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ 0 & k \geq |x| \end{cases}$$

- Intuition: $\texttt{sign\_extend}_n\ (x_{m-1} \cdots x_0) = (\underbrace{x_{m-1} \cdots x_{m-1}}_{n-\text{mal}}\ x_{m-1} \cdots x_0)$

  $|\texttt{sign\_extend}_n\ x| = |x| + n$

  $$(\texttt{sign\_extend}_n\ x)_k = \begin{cases} x_k & 0 \leq k < |x| \\ x_{|x|-1} & k \geq |x| \end{cases}$$

- $\langle b \rangle_U = \langle \texttt{zero\_extend}_n\ b \rangle_U$ $\qquad$ $\langle b \rangle_S = \langle \texttt{sign\_extend}_n\ b \rangle_S$

Semantik 3

- $|\texttt{bvnot}\ x| = |x|$

$$(\texttt{bvnot}\ x)_k = \begin{cases} 0 & x_k = 1 \\ 1 & x_k = 0 \end{cases}$$

- $\texttt{bvand}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvand}\ x\ y| = |x|$$

$$(\texttt{bvand}\ x\ y)_k = \begin{cases} 1 & x_k = y_k = 1 \\ 0 & \text{sonst} \end{cases}$$

- $\texttt{bvor}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvor}\ x\ y| = |x|$$

$$(\texttt{bvor}\ x\ y)_k = \begin{cases} 0 & x_k = y_k = 0 \\ 1 & \text{sonst} \end{cases}$$

Semantik 3

- $|\texttt{bvnot}\ x| = |x|$

$$(\texttt{bvnot}\ x)_k = \begin{cases} 0 & x_k = 1 \\ 1 & x_k = 0 \end{cases}$$

- $\texttt{bvand}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvand}\ x\ y| = |x|$$

$$(\texttt{bvand}\ x\ y)_k = \begin{cases} 1 & x_k = y_k = 1 \\ 0 & \text{sonst} \end{cases}$$

- $\texttt{bvor}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvor}\ x\ y| = |x|$$

$$(\texttt{bvor}\ x\ y)_k = \begin{cases} 0 & x_k = y_k = 0 \\ 1 & \text{sonst} \end{cases}$$

Semantik 3

- $|\texttt{bvnot}\ x| = |x|$

$$(\texttt{bvnot}\ x)_k = \begin{cases} 0 & x_k = 1 \\ 1 & x_k = 0 \end{cases}$$

- $\texttt{bvand}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvand}\ x\ y| = |x|$$

$$(\texttt{bvand}\ x\ y)_k = \begin{cases} 1 & x_k = y_k = 1 \\ 0 & \text{sonst} \end{cases}$$

- $\texttt{bvor}\ x\ y$ ist nur definiert falls $|x| = |y|$

$$|\texttt{bvor}\ x\ y| = |x|$$

$$(\texttt{bvor}\ x\ y)_k = \begin{cases} 0 & x_k = y_k = 0 \\ 1 & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 4

- Intuition: $\texttt{bvshl}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-3} \cdots x_0 00)$

  $\texttt{bvshl}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvshl}\ x\ y| = |x|$

  $$(\texttt{bvshl}\ x\ y)_k = \begin{cases} x_{k-\langle y \rangle_U} & k \geq \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvlshr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (00 x_{n-1} \cdots x_2)$

  $\texttt{bvlshr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvlshr}\ x\ y| = |x|$

  $$(\texttt{bvlshr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ 0 & \text{sonst} \end{cases}$$

- Intuition: $\texttt{bvashr}\ (x_{n-1} \cdots x_0)\ \texttt{bv}_{n,2} = (x_{n-1} x_{n-1} x_{n-1} \cdots x_2)$

  $\texttt{bvashr}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvashr}\ x\ y| = |x|$

  $$(\texttt{bvashr}\ x\ y)_k = \begin{cases} x_{k+\langle y \rangle_U} & k < |x| - \langle y \rangle_U \\ x_{|x|-1} & \text{sonst} \end{cases}$$

Semantik 5

- $\texttt{bvadd}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvadd}\ x\ y| = |x|$

  $\langle \texttt{bvadd}\ x\ y \rangle_U \equiv \langle x \rangle_U + \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvsub}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvsub}\ x\ y| = |x|$

  $\langle \texttt{bvsub}\ x\ y \rangle_U \equiv \langle x \rangle_U - \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvmul}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvmul}\ x\ y| = |x|$

  $\langle \texttt{bvmul}\ x\ y \rangle_U \equiv \langle x \rangle_U * \langle y \rangle_U \mod 2^{|x|}$

Semantik 5

- $\texttt{bvadd}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvadd}\ x\ y| = |x|$

  $\langle \texttt{bvadd}\ x\ y \rangle_U \equiv \langle x \rangle_U + \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvsub}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvsub}\ x\ y| = |x|$

  $\langle \texttt{bvsub}\ x\ y \rangle_U \equiv \langle x \rangle_U - \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvmul}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvmul}\ x\ y| = |x|$

  $\langle \texttt{bvmul}\ x\ y \rangle_U \equiv \langle x \rangle_U * \langle y \rangle_U \mod 2^{|x|}$

Semantik 5

- $\texttt{bvadd}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvadd}\ x\ y| = |x|$

  $\langle \texttt{bvadd}\ x\ y \rangle_U \equiv \langle x \rangle_U + \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvsub}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvsub}\ x\ y| = |x|$

  $\langle \texttt{bvsub}\ x\ y \rangle_U \equiv \langle x \rangle_U - \langle y \rangle_U \mod 2^{|x|}$

- $\texttt{bvmul}\ x\ y$ ist nur definiert falls $|x| = |y|$ und $|\texttt{bvmul}\ x\ y| = |x|$

  $\langle \texttt{bvmul}\ x\ y \rangle_U \equiv \langle x \rangle_U * \langle y \rangle_U \mod 2^{|x|}$

Semantik 6

- $x = y$ ist nur definiert falls $|x| = |y|$

  $(x_{n-1} \cdots x_0) = (y_{n-1} \cdots y_0)$ gdw. $x_i = y_i$ für alle $1 \leq i < n$

- `bvult` x y ist nur definiert falls $|x| = |y|$

  `bvult` x y gdw. $\langle x \rangle_U < \langle y \rangle_U$

- `bvule` x y ist nur definiert falls $|x| = |y|$

  `bvule` x y gdw. $\langle x \rangle_U \leq \langle y \rangle_U$

- `bvslt` x y ist nur definiert falls $|x| = |y|$

  `bvslt` x y gdw. $\langle x \rangle_S < \langle y \rangle_S$

- `bvsle` x y ist nur definiert falls $|x| = |y|$

  `bvsle` x y gdw. $\langle x \rangle_S \leq \langle y \rangle_S$

Semantik 6

- $x = y$ ist nur definiert falls $|x| = |y|$

  $(x_{n-1} \cdots x_0) = (y_{n-1} \cdots y_0)$ gdw. $x_i = y_i$ für alle $1 \leq i < n$

- `bvult` x y ist nur definiert falls $|x| = |y|$

  `bvult` x y gdw. $\langle x \rangle_U < \langle y \rangle_U$

- `bvule` x y ist nur definiert falls $|x| = |y|$

  `bvule` x y gdw. $\langle x \rangle_U \leq \langle y \rangle_U$

- `bvslt` x y ist nur definiert falls $|x| = |y|$

  `bvslt` x y gdw. $\langle x \rangle_S < \langle y \rangle_S$

- `bvsle` x y ist nur definiert falls $|x| = |y|$

  `bvsle` x y gdw. $\langle x \rangle_S \leq \langle y \rangle_S$

Semantik 6

- $x = y$ ist nur definiert falls $|x| = |y|$

  $(x_{n-1} \cdots x_0) = (y_{n-1} \cdots y_0)$ gdw. $x_i = y_i$ für alle $1 \leq i < n$

- `bvult` x y ist nur definiert falls $|x| = |y|$

  `bvult` x y gdw. $\langle x \rangle_U < \langle y \rangle_U$

- `bvule` x y ist nur definiert falls $|x| = |y|$

  `bvule` x y gdw. $\langle x \rangle_U \leq \langle y \rangle_U$

- `bvslt` x y ist nur definiert falls $|x| = |y|$

  `bvslt` x y gdw. $\langle x \rangle_S < \langle y \rangle_S$

- `bvsle` x y ist nur definiert falls $|x| = |y|$

  `bvsle` x y gdw. $\langle x \rangle_S \leq \langle y \rangle_S$

Semantik 6

- $x = y$ ist nur definiert falls $|x| = |y|$

  $(x_{n-1} \cdots x_0) = (y_{n-1} \cdots y_0)$ gdw. $x_i = y_i$ für alle $1 \leq i < n$

- `bvult` x y ist nur definiert falls $|x| = |y|$

  `bvult` x y gdw. $\langle x \rangle_U < \langle y \rangle_U$

- `bvule` x y ist nur definiert falls $|x| = |y|$

  `bvule` x y gdw. $\langle x \rangle_U \leq \langle y \rangle_U$

- `bvslt` x y ist nur definiert falls $|x| = |y|$

  `bvslt` x y gdw. $\langle x \rangle_S < \langle y \rangle_S$

- `bvsle` x y ist nur definiert falls $|x| = |y|$

  `bvsle` x y gdw. $\langle x \rangle_S \leq \langle y \rangle_S$

Semantik 6

- $x = y$ ist nur definiert falls $|x| = |y|$

  $(x_{n-1} \cdots x_0) = (y_{n-1} \cdots y_0)$ gdw. $x_i = y_i$ für alle $1 \leq i < n$

- `bvult` x y ist nur definiert falls $|x| = |y|$

  `bvult` x y gdw. $\langle x \rangle_U < \langle y \rangle_U$

- `bvule` x y ist nur definiert falls $|x| = |y|$

  `bvule` x y gdw. $\langle x \rangle_U \leq \langle y \rangle_U$

- `bvslt` x y ist nur definiert falls $|x| = |y|$

  `bvslt` x y gdw. $\langle x \rangle_S < \langle y \rangle_S$

- `bvsle` x y ist nur definiert falls $|x| = |y|$

  `bvsle` x y gdw. $\langle x \rangle_S \leq \langle y \rangle_S$

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $\texttt{bvslt}(\texttt{bv}_{32,0}, \texttt{bvsub}(x, y)) \wedge \neg\texttt{bvslt}(y, x)$ erfüllbar? JA

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $\texttt{bvslt}(\texttt{bv}_{32,0}, \texttt{bvsub}(x, y)) \wedge \neg\texttt{bvslt}(y, x)$ erfüllbar? JA

Beispiel

Kann die folgende Assertion fehlschlagen?

```
int x, y;
...
if (x - y > 0) {
    assert(x > y);
    ...
}
...
```

Ist $\texttt{bvslt}(\texttt{bv}_{32,0}, \texttt{bvsub}(x, y)) \wedge \neg\texttt{bvslt}(y, x)$ erfüllbar? JA

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Entscheidungsverfahren für Bitvektorlogik

- Erste Möglichkeit: DPLL(T) mit Bitvektorlogik-Solver
  - Konjunktion von Bitvektorlogik wird nach SAT reduziert
- Zweite Möglichkeit: Komplette Formel wird nach SAT reduziert
  - Direkter Ansatz ohne DPLL(T)-Overhead
  - In der Praxis bessere Performance
- Notation: Sei ϕ eine Bitvektorlogikformel
  - $\mathrm{At}(\phi)$: Atome in ϕ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\phi)$
  - $\mathrm{BS}(\phi)$: Boolesches Skelett von ϕ (ersetze $a \in \mathrm{At}(\phi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\phi)$: Terme in ϕ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\phi)$

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel ϕ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:
  1. $\psi := \mathrm{BS}(\phi)$
  2. Für jedes $a \in \mathrm{At}(\phi)$:

     $\psi := \psi \wedge \text{BV-Constraint}(a)$
  3. Für jedes $t \in \mathrm{T}(\phi)$:

     $\psi := \psi \wedge \text{BV-Constraint}(t)$
  4. Gib ψ aus

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel ϕ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:
  1. $\psi := \mathrm{BS}(\phi)$
  2. Für jedes $a \in \mathrm{At}(\phi)$:

     $$\psi := \psi \wedge \text{BV-Constraint}(a)$$

  3. Für jedes $t \in \mathrm{T}(\phi)$:

     $$\psi := \psi \wedge \text{BV-Constraint}(t)$$

  4. Gib ψ aus

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel ϕ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:
  1. $\psi := \mathrm{BS}(\phi)$
  2. Für jedes $a \in \mathrm{At}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(a)$$

  3. Für jedes $t \in \mathrm{T}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(t)$$

  4. Gib ψ aus

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel ϕ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:
  1. $\psi := \mathrm{BS}(\phi)$
  2. Für jedes $a \in \mathrm{At}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(a)$$

  3. Für jedes $t \in \mathrm{T}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(t)$$

  4. Gib ψ aus

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel ϕ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:
  1. $\psi := \mathrm{BS}(\phi)$
  2. Für jedes $a \in \mathrm{At}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(a)$$

  3. Für jedes $t \in \mathrm{T}(\phi)$:

$$\psi := \psi \wedge \text{BV-Constraint}(t)$$

  4. Gib ψ aus

BV-Constraint(a) 1

- BV-Constraint($x = y$):

$$\mathrm{AV}(x = y) \quad \leftrightarrow \quad \bigwedge_{i=0}^{|x|-1} \mathrm{E}(x)_i \leftrightarrow \mathrm{E}(y)_i$$

- BV-Constraint(`bvult` x y):

$$\mathrm{AV}(\texttt{bvult}\ x\ y) \quad \leftrightarrow \quad \bigvee_{i=0}^{|x|-1} \left(\neg \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i \wedge \left(\bigwedge_{j=i+1}^{|x|-1} \mathrm{E}(x)_j \leftrightarrow \mathrm{E}(y)_j \right) \right)$$

- BV-Constraint(`bvule` x y):

$$\mathrm{AV}(\texttt{bvule}\ x\ y) \quad \leftrightarrow \quad \ldots$$

BV-Constraint(a) 1

- BV-Constraint($x = y$):

$$\mathrm{AV}(x = y) \quad \leftrightarrow \quad \bigwedge_{i=0}^{|x|-1} \mathrm{E}(x)_i \leftrightarrow \mathrm{E}(y)_i$$

- BV-Constraint($\texttt{bvult}\ x\ y$):

$$\mathrm{AV}(\texttt{bvult}\ x\ y) \quad \leftrightarrow \quad \bigvee_{i=0}^{|x|-1} \left(\neg \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i \wedge \left(\bigwedge_{j=i+1}^{|x|-1} \mathrm{E}(x)_j \leftrightarrow \mathrm{E}(y)_j \right) \right)$$

- BV-Constraint($\texttt{bvule}\ x\ y$):

$$\mathrm{AV}(\texttt{bvule}\ x\ y) \quad \leftrightarrow \quad \ldots$$

BV-Constraint(a) 1

- BV-Constraint($x = y$):

$$\mathrm{AV}(x = y) \quad \leftrightarrow \quad \bigwedge_{i=0}^{|x|-1} \mathrm{E}(x)_i \leftrightarrow \mathrm{E}(y)_i$$

- BV-Constraint(`bvult` x y):

$$\mathrm{AV}(\texttt{bvult}\ x\ y) \quad \leftrightarrow \quad \bigvee_{i=0}^{|x|-1} \left(\neg \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i \wedge \left(\bigwedge_{j=i+1}^{|x|-1} \mathrm{E}(x)_j \leftrightarrow \mathrm{E}(y)_j \right) \right)$$

- BV-Constraint(`bvule` x y):

$$\mathrm{AV}(\texttt{bvule}\ x\ y) \quad \leftrightarrow \quad \ldots$$

BV-Constraint(a) 2

- BV-Constraint(`bvslt` x y):

$$\mathrm{AV}(\texttt{bvslt}\ x\ y) \quad \leftrightarrow \quad (\mathrm{E}(x)_{|x|-1} \wedge \neg \mathrm{E}(y)_{|x|-1})$$
$$\vee\ [\ \mathrm{E}(x)_{|x|-1} \leftrightarrow \mathrm{E}(y)_{|x|-1} \wedge$$
$$\bigvee_{i=0}^{|x|-2} \left(\neg \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i \wedge \left(\bigwedge_{j=i+1}^{|x|-2} \mathrm{E}(x)_j \leftrightarrow \mathrm{E}(y)_j \right) \right)]$$

- BV-Constraint(`bvsle` x y):

$$\mathrm{AV}(\texttt{bvsle}\ x\ y) \quad \leftrightarrow \quad \ldots$$

BV-Constraint(a) 2

- BV-Constraint(`bvslt` x y):

$$\mathrm{AV}(\texttt{bvslt}\ x\ y) \quad \leftrightarrow \quad (\mathrm{E}(x)_{|x|-1} \wedge \neg \mathrm{E}(y)_{|x|-1})$$
$$\vee\, [\, \mathrm{E}(x)_{|x|-1} \leftrightarrow \mathrm{E}(y)_{|x|-1} \wedge$$
$$\bigvee_{i=0}^{|x|-2} \left(\neg \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i \wedge \left(\bigwedge_{j=i+1}^{|x|-2} \mathrm{E}(x)_j \leftrightarrow \mathrm{E}(y)_j \right) \right)]$$

- BV-Constraint(`bvsle` x y):

$$\mathrm{AV}(\texttt{bvsle}\ x\ y) \quad \leftrightarrow \quad \ldots$$

BV-Constraint(t) 1

- BV-Constraint(x) für Variablen x:

$$\top$$

- BV-Constraint($\mathrm{bv}_{n,k}$):
  sei $\langle (b_{n-1} \cdots b_0) \rangle_U = k$

$$\bigwedge_{i=0}^{n-1} \mathrm{E}(\mathrm{bv}_{n,k})_i \leftrightarrow b_i$$

- BV-Constraint($\texttt{concat}\ x\ y$):

$$\bigwedge_{i=|y|}^{|x|+|y|-1} \mathrm{E}(\texttt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_{i-|y|} \wedge \bigwedge_{i=0}^{|y|-1} \mathrm{E}(\texttt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(y)_i$$

BV-Constraint(t) 1

- BV-Constraint(x) für Variablen x:

$$\top$$

- BV-Constraint($\mathtt{bv}_{n,k}$):
  sei $\langle (b_{n-1} \cdots b_0) \rangle_U = k$

$$\bigwedge_{i=0}^{n-1} \mathrm{E}(\mathtt{bv}_{n,k})_i \leftrightarrow b_i$$

- BV-Constraint($\mathtt{concat}\ x\ y$):

$$\bigwedge_{i=|y|}^{|x|+|y|-1} \mathrm{E}(\mathtt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_{i-|y|} \wedge \bigwedge_{i=0}^{|y|-1} \mathrm{E}(\mathtt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(y)_i$$

BV-Constraint(t) 1

- BV-Constraint(x) für Variablen x:

$$\top$$

- BV-Constraint($\mathtt{bv}_{n,k}$):
  sei $\langle (b_{n-1} \cdots b_0) \rangle_U = k$

$$\bigwedge_{i=0}^{n-1} \mathrm{E}(\mathtt{bv}_{n,k})_i \leftrightarrow b_i$$

- BV-Constraint($\mathtt{concat}\ x\ y$):

$$\bigwedge_{i=|y|}^{|x|+|y|-1} \mathrm{E}(\mathtt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_{i-|y|} \wedge \bigwedge_{i=0}^{|y|-1} \mathrm{E}(\mathtt{concat}\ x\ y)_i \leftrightarrow \mathrm{E}(y)_i$$

BV-Constraint(t) 2

- BV-Constraint($\texttt{extract}_{i,j}\ x$):

$$\bigwedge_{k=0}^{i-j+1} \mathrm{E}(\texttt{extract}_{i,j})_k \leftrightarrow \mathrm{E}(x)_{j+k}$$

- BV-Constraint($\texttt{zero\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \neg\mathrm{E}(\texttt{zero\_extend}_n\ x)_k \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{zero\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

- BV-Constraint($\texttt{sign\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_{|x|-1} \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

BV-Constraint(t) 2

- BV-Constraint($\texttt{extract}_{i,j}\ x$):

$$\bigwedge_{k=0}^{i-j+1} \mathrm{E}(\texttt{extract}_{i,j})_k \leftrightarrow \mathrm{E}(x)_{j+k}$$

- BV-Constraint($\texttt{zero\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \neg\mathrm{E}(\texttt{zero\_extend}_n\ x)_k \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{zero\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

- BV-Constraint($\texttt{sign\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_{|x|-1} \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

BV-Constraint(t) 2

- BV-Constraint($\texttt{extract}_{i,j}\ x$):

$$\bigwedge_{k=0}^{i-j+1} \mathrm{E}(\texttt{extract}_{i,j})_k \leftrightarrow \mathrm{E}(x)_{j+k}$$

- BV-Constraint($\texttt{zero\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \neg\mathrm{E}(\texttt{zero\_extend}_n\ x)_k \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{zero\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

- BV-Constraint($\texttt{sign\_extend}_n\ x$):

$$\bigwedge_{k=|x|}^{|x|+n-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_{|x|-1} \wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{sign\_extend}_n\ x)_k \leftrightarrow \mathrm{E}(x)_k$$

BV-Constraint(t) 3

- BV-Constraint($\texttt{bvnot}\ x$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvnot}\ x)_i \leftrightarrow \neg \mathrm{E}(x)_i$$

- BV-Constraint($\texttt{bvand}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvand}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i$$

- BV-Constraint($\texttt{bvor}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvor}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \vee \mathrm{E}(y)_i$$

BV-Constraint(t) 3

- BV-Constraint(`bvnot` x):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvnot}\ x)_i \leftrightarrow \neg \mathrm{E}(x)_i$$

- BV-Constraint(`bvand` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvand}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i$$

- BV-Constraint(`bvor` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvor}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \vee \mathrm{E}(y)_i$$

BV-Constraint(t) 3

- BV-Constraint($\texttt{bvnot}\ x$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvnot}\ x)_i \leftrightarrow \neg \mathrm{E}(x)_i$$

- BV-Constraint($\texttt{bvand}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvand}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \wedge \mathrm{E}(y)_i$$

- BV-Constraint($\texttt{bvor}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvor}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_i \vee \mathrm{E}(y)_i$$

BV-Constraint(t) 4

- BV-Constraint(`bvshl` x y):

$$(\text{“}\texttt{bvult}\ y\ \texttt{bv}_{|x|,|x|}\text{”} \wedge \bigvee_{i=0}^{|x|-1} \mathrm{ls}(x, y, i))$$

$$\vee\ (\text{“}\texttt{bvule}\ \texttt{bv}_{|x|,|x|}\ y\text{”} \wedge \bigwedge_{i=0}^{|x|-1} \neg \mathrm{E}(\texttt{bvshl}\ x\ y)_i))$$

$$\mathrm{ls}(x, y, i) := \text{“}y = \texttt{bv}_{|x|,i}\text{”}$$
$$\wedge \bigwedge_{k=0}^{i-1} \neg \mathrm{E}(\texttt{bvshl}\ x\ y)_k$$
$$\wedge \bigwedge_{k=0}^{|x|-1} \mathrm{E}(\texttt{bvshl}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{k-i}$$

BV-Constraint(t) 4

- BV-Constraint($\texttt{bvshl}\ x\ y$):

$$(\text{“}\texttt{bvult}\ y\ \texttt{bv}_{|x|,|x|}\text{”} \wedge \bigvee_{i=0}^{|x|-1} \mathrm{ls}(x, y, i))$$

$$\vee\ (\text{“}\texttt{bvule}\ \texttt{bv}_{|x|,|x|}\ y\text{”} \wedge \bigwedge_{i=0}^{|x|-1} \neg \mathrm{E}(\texttt{bvshl}\ x\ y)_i))$$

$$\begin{aligned}\mathrm{ls}(x, y, i) := \quad & \text{“}y = \texttt{bv}_{|x|,i}\text{”} \\ & \wedge \bigwedge_{k=0}^{i-1} \neg \mathrm{E}(\texttt{bvshl}\ x\ y)_k \\ & \wedge \bigwedge_{k=i}^{|x|-1} \mathrm{E}(\texttt{bvshl}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{k-i}\end{aligned}$$

BV-Constraint(t) 5

- BV-Constraint($\texttt{bvlshr}\ x\ y$):

$$(\text{“}\texttt{bvult}\ y\ \texttt{bv}_{|x|,|x|}\text{”} \wedge \bigvee_{i=0}^{|x|-1} \mathrm{lsr}(x, y, i))$$

$$\vee\ (\text{“}\texttt{bvule}\ \texttt{bv}_{|x|,|x|}\ y\text{”} \wedge \bigwedge_{i=0}^{|x|-1} \neg \mathrm{E}(\texttt{bvlshr}\ x\ y)_i)$$

$$\mathrm{lsr}(x, y, i) := \quad \text{“}y = \texttt{bv}_{|x|,i}\text{”}$$
$$\wedge \bigwedge_{k=0}^{|x|-i-1} \mathrm{E}(\texttt{bvlshr}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{k+i}$$
$$\wedge \bigwedge_{k=0}^{|x|-1} \neg \mathrm{E}(\texttt{bvlshr}\ x\ y)_k$$

BV-Constraint(t) 5

- BV-Constraint(`bvlshr` x y):

$$(\text{“}\texttt{bvult}\ y\ \texttt{bv}_{|x|,|x|}\text{”} \wedge \bigvee_{i=0}^{|x|-1} \mathrm{lsr}(x, y, i))$$

$$\vee\ (\text{“}\texttt{bvule}\ \texttt{bv}_{|x|,|x|}\ y\text{”} \wedge \bigwedge_{i=0}^{|x|-1} \neg \mathrm{E}(\texttt{bvlshr}\ x\ y)_i)$$

$$\begin{aligned}\mathrm{lsr}(x, y, i) := \quad & \text{“}y = \texttt{bv}_{|x|,i}\text{”} \\ & \wedge \bigwedge_{k=0}^{|x|-i-1} \mathrm{E}(\texttt{bvlshr}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{k+i} \\ & \wedge \bigwedge^{|x|-1} \neg \mathrm{E}(\texttt{bvlshr}\ x\ y)_k\end{aligned}$$

BV-Constraint(t) 6

- BV-Constraint($\texttt{bvashr}\ x\ y$):

$$(\text{“}\texttt{bvult}\ y\ \texttt{bv}_{|x|,|x|}\text{”} \wedge \bigvee_{i=0}^{|x|-1} \mathrm{asr}(x,y,i))$$

$$\vee\ (\text{“}\texttt{bvule}\ \texttt{bv}_{|x|,|x|}\ y\text{”} \wedge \bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvashr}\ x\ y)_i \leftrightarrow \mathrm{E}(x)_{|x|-1})$$

$$\begin{aligned}\mathrm{asr}(x,y,i) := \quad & \text{“}y = \texttt{bv}_{|x|,i}\text{”} \\ & \wedge \bigwedge_{k=0}^{|x|-i-1} \mathrm{E}(\texttt{bvashr}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{k+i} \\ & \wedge \bigwedge_{k=|x|-i-1}^{|x|-1} \mathrm{E}(\texttt{bvashr}\ x\ y)_k \leftrightarrow \mathrm{E}(x)_{|x|-1}\end{aligned}$$

BV-Constraint(t) 7

Definition (Volladdierer)

Für Bits a, b, c_{in}:

$$\text{sum}(a, b, c_{\text{in}}) = a \oplus b \oplus c_{\text{in}}$$

$$\text{carry}(a, b, c_{\text{in}}) = (a \wedge b) \vee (a \wedge c_{\text{in}}) \vee (b \wedge c_{\text{in}})$$

Definition (Übertragsbits)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$c_i = \begin{cases} c_{\text{in}} & i = 0 \\ \text{carry}(x_{i-1}, y_{i-1}, c_{i-1}) & i > 0 \end{cases}$$

Definition (Addierwerk)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$\text{add}(x, y, c_{\text{in}}) = (r_{|x|-1} \cdots r_0)$$

$$r_i = \text{sum}(x_i, y_i, c_i)$$

BV-Constraint(t) 7

Definition (Volladdierer)

Für Bits a, b, c_{in}:

$$\text{sum}(a, b, c_{\text{in}}) = a \oplus b \oplus c_{\text{in}}$$

$$\text{carry}(a, b, c_{\text{in}}) = (a \wedge b) \vee (a \wedge c_{\text{in}}) \vee (b \wedge c_{\text{in}})$$

Definition (Übertragsbits)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$c_i = \begin{cases} c_{\text{in}} & i = 0 \\ \text{carry}(x_{i-1}, y_{i-1}, c_{i-1}) & i > 0 \end{cases}$$

Definition (Addierwerk)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$\text{add}(x, y, c_{\text{in}}) = (r_{|x|-1} \cdots r_0)$$

$$r_i = \text{sum}(x_i, y_i, c_i)$$

BV-Constraint(t) 7

Definition (Volladdierer)

Für Bits a, b, c_{in}:

$$\text{sum}(a, b, c_{\text{in}}) = a \oplus b \oplus c_{\text{in}}$$

$$\text{carry}(a, b, c_{\text{in}}) = (a \wedge b) \vee (a \wedge c_{\text{in}}) \vee (b \wedge c_{\text{in}})$$

Definition (Übertragsbits)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$c_i = \begin{cases} c_{\text{in}} & i = 0 \\ \text{carry}(x_{i-1}, y_{i-1}, c_{i-1}) & i > 0 \end{cases}$$

Definition (Addierwerk)

Für Listen x, y der Länge n von Bits und ein Bit c_{in}:

$$\text{add}(x, y, c_{\text{in}}) = (r_{|x|-1} \cdots r_0)$$

$$r_i = \text{sum}(x_i, y_i, c_i)$$

BV-Constraint(t) 8

- BV-Constraint($\texttt{bvadd}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvadd}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \mathrm{E}(y), \bot)_i$$

- BV-Constraint($\texttt{bvsub}\ x\ y$):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvsub}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \text{“}\texttt{bvnot}\ y\text{”}, \top)_i$$

- $\texttt{bvmul}$ kann mittels $\texttt{extract}_{i,j}$, $\texttt{bvshl}$ und $\texttt{bvadd}$ implementiert werden

BV-Constraint(t) 8

- BV-Constraint(`bvadd` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvadd}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \mathrm{E}(y), \bot)_i$$

- BV-Constraint(`bvsub` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvsub}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \text{“}\texttt{bvnot}\ y\text{”}, \top)_i$$

- `bvmul` kann mittels $\texttt{extract}_{i,j}$, `bvshl` und `bvadd` implementiert werden

BV-Constraint(t) 8

- BV-Constraint(`bvadd` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvadd}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \mathrm{E}(y), \bot)_i$$

- BV-Constraint(`bvsub` x y):

$$\bigwedge_{i=0}^{|x|-1} \mathrm{E}(\texttt{bvsub}\ x\ y)_i \leftrightarrow \mathrm{add}(\mathrm{E}(x), \text{“}\texttt{bvnot}\ y\text{”}, \top)_i$$

- `bvmul` kann mittels $\texttt{extract}_{i,j}$, `bvshl` und `bvadd` implementiert werden

Inkrementelles Bit-Blasting

- Einige Operationen sind “teuer”
- `bvmul` für n Bits:

n	SAT-Variablen	Klauseln
8	313	1001
16	1265	4177
32	5089	17057
64	20417	68929

- Idee: BV-Constraint für “teure” Operationen wird nur hinzugefügt falls die Formel ohne diese Operationen erfüllbar ist
- Alternative: Approximiere die “teuren” Operationen im ersten Schritt durch uninterpretierte Funktionen
  - Benutze Ackermanns Reduktion um die uninterpretierten Funktionen durch Variablen zu ersetzen

Inkrementelles Bit-Blasting

- Einige Operationen sind “teuer”
- `bvmul` für n Bits:

n	SAT-Variablen	Klauseln
8	313	1001
16	1265	4177
32	5089	17057
64	20417	68929

- Idee: BV-Constraint für “teure” Operationen wird nur hinzugefügt falls die Formel ohne diese Operationen erfüllbar ist
- Alternative: Approximiere die “teuren” Operationen im ersten Schritt durch uninterpretierte Funktionen
  - Benutze Ackermanns Reduktion um die uninterpretierten Funktionen durch Variablen zu ersetzen

Inkrementelles Bit-Blasting

- Einige Operationen sind “teuer”
- `bvmul` für n Bits:

n	SAT-Variablen	Klauseln
8	313	1001
16	1265	4177
32	5089	17057
64	20417	68929

- Idee: $\mathrm{BV\text{-}Constraint}$ für “teure” Operationen wird nur hinzugefügt falls die Formel ohne diese Operationen erfüllbar ist
- Alternative: Approximiere die “teuren” Operationen im ersten Schritt durch uninterpretierte Funktionen
  - Benutze Ackermanns Reduktion um die uninterpretierten Funktionen durch Variablen zu ersetzen

Inkrementelles Bit-Blasting

- Einige Operationen sind “teuer”
- `bvmul` für n Bits:

n	SAT-Variablen	Klauseln
8	313	1001
16	1265	4177
32	5089	17057
64	20417	68929

- Idee: $\mathrm{BV\text{-}Constraint}$ für “teure” Operationen wird nur hinzugefügt falls die Formel ohne diese Operationen erfüllbar ist
- Alternative: Approximiere die “teuren” Operationen im ersten Schritt durch uninterpretierte Funktionen
  - Benutze Ackermanns Reduktion um die uninterpretierten Funktionen durch Variablen zu ersetzen