Bitvektoren

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

STEPHAN FALKE — INSTITUT FÜR THEORETISCHE INFORMATIK (ITI)

Inhalt

1. Wieso Bitvektoren?
2. Entscheidungsverfahren für Bitvektoren

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

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

Bitvektorlogik

Theorie

- Funktionssymbole:

  $(\mathtt{bv}_{n,k})_{n>0 \wedge 0 \leq k \leq 2^n - 1}$

  $\mathtt{concat}$, $(\mathtt{extract}_{i,j})_{0 \leq j < i}$

  $(\mathtt{zero\_extend}_n)_{n>0}$, $(\mathtt{sign\_extend}_n)_{n>0}$

  $\mathtt{bvnot}$, $\mathtt{bvand}$, $\mathtt{bvor}$

  $\mathtt{bvshl}$, $\mathtt{bvlshr}$, $\mathtt{bvashr}$

  $\mathtt{bvadd}$, $\mathtt{bvsub}$, $\mathtt{bvmul}$

  $\ldots$

- Prädikatessymbole:

  $=$, $\mathtt{bvult}$, $\mathtt{bvule}$, $\mathtt{bvslt}$, $\mathtt{bvsle}$

- 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

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

Semantik 1

- $\left|\texttt{bv}_{n,k}\right| = n$

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

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

  $\left|\texttt{concat}\ x\ y\right| = |x| + |y|$

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

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

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

  $\left|\texttt{extract}_{i,j}\ x\right| = i - j + 1$

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

Semantik 2

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

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

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

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

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

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

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

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

  $\texttt{bvlshr}\ x\ y$ 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 $|\texttt{bvadd}\ x\ y| = |x|$

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

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

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

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

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

Semantik 6

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

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

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

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

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

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

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

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

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

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

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

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
  - $\mathrm{At}(\varphi)$: Atome in φ
  - $\mathrm{AV}(a)$: Abstraktionsvariable für $a \in \mathrm{At}(\varphi)$
  - $\mathrm{BS}(\varphi)$: Boolesches Skelett von φ (ersetze $a \in \mathrm{At}(\varphi)$ durch $\mathrm{AV}(a)$)
  - $\mathrm{T}(\varphi)$: Terme in φ
  - $\mathrm{E}(t)$: Liste der Länge $|t|$ von SAT-Variablen (Bits) für $t \in \mathrm{T}(\varphi)$

Bit-Blasting

Algorithmus

- Eingabe: Bitvektorlogikformel φ
- Ausgabe: Erfüllbarkeitsäquivalente SAT-Formel
- Schritte:

1. $\psi := \mathrm{BS}(\varphi)$
2. Für jedes $a \in \mathrm{At}(\varphi)$:

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

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

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

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$(\texttt{bvult}\ x\ y)$:

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

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

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

BV-Constraint(a) 2

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

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

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

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

BV-Constraint(t) 1

- BV-Constraint(x) für Variablen x:

$$\top$$

- BV-Constraint$(\mathtt{bv}_{n,k})$:
  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) 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$(\texttt{bvshl}\ x\ y)$:

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

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

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

BV-Constraint(t) 5

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

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

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

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

BV-Constraint(t) 6

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

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

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

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

BV-Constraint(t) 7

Definition (Volladdierer)
Für Bits a, b, c_{in}:

$$\begin{aligned} \mathrm{sum}(a, b, c_{\text{in}}) &= a \oplus b \oplus c_{\text{in}} \\ \mathrm{carry}(a, b, c_{\text{in}}) &= (a \wedge b) \vee (a \wedge c_{\text{in}}) \vee (b \wedge c_{\text{in}}) \end{aligned}$$

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

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

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

$$\begin{aligned} \mathrm{add}(x, y, c_{\text{in}}) &= (r_{|x|-1} \cdots r_0) \\ r_i &= \mathrm{sum}(x_i, y_i, c_i) \end{aligned}$$

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$(\texttt{bvsub}\ x\ y)$:

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

- `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

Reduktion auf LIA 1

- Bitvektorlogik kann auf linear Arithmetik ganzer Zahlen (LIA) reduziert werden
- Idee: Benutze $\langle b \rangle_U$ anstelle des Bitvektor b

Definition

Eine Bitvektorlogikformel φ ist shift-konstant wenn das zweite Argument von jedem Vorkommen von `bvshl`, `bvlshr` und `bvashr` in φ die Form $\texttt{bv}_{n,k}$ hat.

- `bvshl`, `bvlshr` und `bvashr` mit nicht-konstantem zweiten Argument y kann mittels einer Fallunterscheidung behandelt werden
  - $y = \texttt{bv}_{n,0}$

    $\vdots$
  - $y = \texttt{bv}_{n,n-1}$
  - $\texttt{bvule}\ \texttt{bv}_{n,n}\ y$

Reduktion auf LIA 2

- $\mathrm{flat}(\varphi)$ bezeichnet die flache Form von φ
  - Keine geschachtelten Terme
    - Terme sind Variablen, Konstanten oder habe die Form $f(a_1, \ldots, a_n)$ für Variablen oder Konstanten $a_1, \ldots, a_n$
  - $f(g(x), h(y))$ wird ersetzt durch $f(x_1, x_2)$, neue Atome $x_1 = g(x)$ und $x_2 = h(y)$ werden zur Formel hinzugefügt
  - φ und $\mathrm{flat}(\varphi)$ sind erfüllbarkeitsäquivalent

Reduktion auf LIA 3

Algorithmus

- Eingabe: Shift-konstante Bitvektorlogikformel φ
- Ausgabe: Erfüllbarkeitsäquivalente LIA-Formel
- Schritte:
  1. $\varphi' := \mathrm{flat}(\varphi)$
     $\psi := \varphi'$
  2. Für jedes Atom $a \in \mathrm{At}(\varphi')$:

     Sei $(\vartheta, \gamma) := \mathrm{bv2lia}(a)$ in:

     $$\psi := \psi[a/\vartheta] \wedge \gamma$$
  3. Gib ψ aus

$\mathrm{bv2lia}(a)$ 1

- $\mathrm{bv2lia}(x = y)$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  $$\vartheta : x' = y'$$
  $$\gamma : \gamma_x \wedge \gamma_y$$
- $\mathrm{bv2lia}(\texttt{bvult}\ x\ y)$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  $$\vartheta : x' < y'$$
  $$\gamma : \gamma_x \wedge \gamma_y$$
- $\mathrm{bv2lia}(\texttt{bvule}\ x\ y)$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  $$\vartheta : x' \leq y'$$
  $$\gamma : \gamma_x \wedge \gamma_y$$

bv2lia(a) 2

- bv2lia(`bvslt` x y):
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$

$$\begin{aligned} \vartheta: \quad & (x' < 2^{|x|-1} \wedge y' < 2^{|x|-1} \wedge x' < y') \\ \vee\, & (x' \geq 2^{|x|-1} \wedge y' \geq 2^{|x|-1} \wedge x' < y') \\ \vee\, & (x' \geq 2^{|x|-1} \wedge y' < 2^{|x|-1}) \end{aligned}$$

$$\gamma : \gamma_x \wedge \gamma_y$$

- bv2lia(`bvsle` x y):
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$

$$\begin{aligned} \vartheta: \quad & (x' < 2^{|x|-1} \wedge y' < 2^{|x|-1} \wedge x' \leq y') \\ \vee\, & (x' \geq 2^{|x|-1} \wedge y' \geq 2^{|x|-1} \wedge x' \leq y') \\ \vee\, & (x' \geq 2^{|x|-1} \wedge y' < 2^{|x|-1}) \end{aligned}$$

$$\gamma : \gamma_x \wedge \gamma_y$$

$\mathbf{bv2lia}(t)$ 1

- $\mathrm{bv2lia}(x)$ für Variablen x:

$$\vartheta : X$$
$$\gamma : 0 \leq X \wedge X < 2^{|x|}$$

- $\mathrm{bv2lia}(\mathtt{bv}_{n,k})$:

$$\vartheta : k$$
$$\gamma : \top$$

- $\mathrm{bv2lia}(\mathtt{concat}\ x\ y)$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \vartheta_y) = \mathrm{bv2lia}(y)$

$$\vartheta : 2^{|y|} * x' + y'$$
$$\gamma : \top$$

$\mathrm{bv2lia}(t)$ 2

- $\mathrm{bv2lia}(\mathtt{extract}_{i,j}\ x)$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$
  seien X_h, X_m, X_l frische Variablen

$$\vartheta : X_m$$

$$\begin{aligned}\gamma :\quad & x' = 2^{i+1} * X_h + 2^j * X_m + X_l \\ & \land 0 \leq X_l \land X_l < 2^j \\ & \land 0 \leq X_m \land X_m < 2^{i-j+1} \\ & \land 0 \leq X_h \land X_h < 2^{|x|-i-1}\end{aligned}$$

$\mathrm{bv2lia}(t)$ 3

- $\mathrm{bv2lia}(\texttt{zero\_extend}_n\ x)$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$

$$\begin{aligned} \vartheta &: x' \\ \gamma &: \top \end{aligned}$$

- $\mathrm{bv2lia}(\texttt{sign\_extend}_n\ x)$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$
  sei R eine frische Variable

$$\begin{aligned} \vartheta &: R \\ \gamma &: \quad (x' < 2^{|x|-1} \land R = x') \\ &\quad \lor (x' \geq 2^{|x|-1} \land R = x' + 2^{|x|+n} - 2^{|x|}) \end{aligned}$$

$\mathrm{bv2lia}(t)$ 4

- $\mathrm{bv2lia}(\texttt{bvnot}\ x)$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$

$$\vartheta : 2^{|x|} - 1 - x'$$
$$\gamma : \top$$

- $\mathrm{bv2lia}(\texttt{bvand}\ x\ y / \texttt{bvor}\ x\ y)$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  seien $R, R_{|x|-1}, \ldots, R_0, X_{|x|-1}, \ldots, X_0, Y_{|x|-1}, \ldots, Y_0$ frische Variablen

$$\vartheta : R$$
$$\gamma : \quad (R_{|x|-1} = 0 \vee R_{|x|-1} = 1) \wedge \ldots \wedge (Y_0 = 0 \vee Y_0 = 1)$$
$$\wedge\, R = \sum_{i=0}^{|x|-1} 2^i * R_i \wedge x' = \sum_{i=0}^{|x|-1} 2^i * X_i \wedge y' = \sum_{i=0}^{|x|-1} 2^i * Y_i$$
$$\wedge \bigwedge_{i=0}^{|x|-1} R_i = 1 \leftrightarrow (X_i = 1 \circ Y_i = 1)$$

Wobei $\circ = \wedge$ bzw. $\circ = \vee$

$\mathrm{bv2lia}(t)$ 5

- `bvshl` x $\mathtt{bv}_{n,k}$ mit $k < n$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$
  seien X_h, X_l frische Variablen

$$\vartheta : 2^k * X_l$$

$$\begin{aligned} \gamma : \quad & x' = 2^{|x|-k} * X_h + X_l \\ & \wedge 0 \leq X_l \wedge X_l < 2^{|x|-k} \\ & \wedge 0 \leq X_h \wedge X_h < 2^k \end{aligned}$$

- `bvshl` x $\mathtt{bv}_{n,k}$ mit $k \geq n$:

$$\vartheta : 0$$

$$\gamma : \top$$

$\mathrm{bv2lia}(t)$ 6

- $\texttt{bvlshr}\ x\ \texttt{bv}_{n,k}$ mit $k < n$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$
  seien X_h, X_l frische Variablen

$$\begin{aligned}
\vartheta &: X_h \\
\gamma &: \quad x' = 2^k * X_h + X_l \\
&\quad \wedge 0 \leq X_l \wedge X_l < 2^k \\
&\quad \wedge 0 \leq X_h \wedge X_h < 2^{|x|-k}
\end{aligned}$$

- $\texttt{bvlshr}\ x\ \texttt{bv}_{n,k}$ mit $k \geq n$:

$$\begin{aligned}
\vartheta &: 0 \\
\gamma &: \top
\end{aligned}$$

- $\texttt{bvashr}$ kann mittels $\texttt{extract}_{i,j}$ und $\texttt{concat}$ implementiert werden

$\mathrm{bv2lia}(t)$ 7

- $\texttt{bvadd}\ x\ y$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  sei R eine frische Variable

$$\vartheta : R$$
$$\gamma : \quad (x' + y' < 2^{|x|} \wedge R = x' + y') \vee (x' + y' \geq 2^{|x|} \wedge R = x' + y' - 2^{|x|})$$

- $\texttt{bvsub}\ x\ y$:
  seien $(x', \gamma_x) = \mathrm{bv2lia}(x)$ und $(y', \gamma_y) = \mathrm{bv2lia}(y)$
  sei R eine frische Variable

$$\vartheta : R$$
$$\gamma : \quad (x' - y' \geq 0 \wedge R = x' - y') \vee (x' - y' < 0 \wedge R = x' - y' + 2^{|x|})$$

$\mathrm{bv2lia}(t)$ 8

- `bvmul` x $\texttt{bv}_{n,k}$ für $k \neq 0$:
  sei $(x', \gamma_x) = \mathrm{bv2lia}(x)$
  seien R, σ frische Variablen

$$
\begin{aligned}
\vartheta &: R \\
\gamma &: \quad R = k * x' - 2^{|x|} * \sigma \\
&\quad \wedge 0 \leq R \wedge R < 2^{|x|} \\
&\quad \wedge 0 \leq \sigma \wedge \sigma < k
\end{aligned}
$$

- `bvmul` x $\texttt{bv}_{n,k}$ für $k = 0$:

$$
\begin{aligned}
\vartheta &: 0 \\
\gamma &: \top
\end{aligned}
$$

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