Selected Solutions 10 | June 7, 2019

1 The simplest bit blasting transformations for the signed comparisons $\geq_s$ and $>_s$ check for the sign bit (which is 1 for negative numbers), and afterwards relies on the respective unsigned comparisons:

$$\mathbf{B}_r(\mathbf{x}_{k+1} \geq_s \mathbf{y}_{k+1}) = (\neg x_k \wedge y_k) \vee ((x_k \leftrightarrow y_k) \wedge \mathbf{B}(\mathbf{x}[k-1{:}0] \geq_u \mathbf{y}[k-1{:}0]))$$
$$\mathbf{B}_r(\mathbf{x}_{k+1} >_s \mathbf{y}_{k+1}) = (\neg x_k \wedge y_k) \vee ((x_k \leftrightarrow y_k) \wedge \mathbf{B}(\mathbf{x}[k-1{:}0] >_u \mathbf{y}[k-1{:}0]))$$

2 (a) The replacement is incorrect. For example, the SMT encoding

```
(declare-const x (_ BitVec 8))
(declare-const c1 (_ BitVec 8))
(declare-const c2 (_ BitVec 8))
(declare-const before (_ BitVec 8))
(declare-const after (_ BitVec 8))
(assert (= before (bvudiv (bvlshr x c1) c2)))
(assert (= after (bvudiv x (bvshl c1 c2))))
(assert (not (= before after)))
(check-sat)
(get-model)
```

shows that for c1 $= \mathbf{x01}_8$, c2 $= \mathbf{x05}_8$, and x $= \mathbf{x44}_8$ the left-hand side evaluates to $\mathbf{x06}_8$ while the right-hand side evaluates to $\mathbf{x02}_8$. A counterexample is also found when the bit vector size is changed to 16.

(b) The replacement is incorrect. For example, the SMT encoding

```
(declare-const p (_ BitVec 8))
(declare-const x (_ BitVec 8))
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const before (_ BitVec 8))
(declare-const after (_ BitVec 8))
(define-fun is-power-of-two ((x (_ BitVec 8))) Bool
  (= #x00 (bvand x (bvsub x #x01))))
(assert (is-power-of-two p))
(assert (= before (bvudiv x (bvlshr (bvshl p a) b))))
(assert (= after (bvudiv x (bvshl p (bvsub a b)))))
(assert (not (= before after)))
(check-sat)
(get-model)
```

shows that for a $= \mathbf{x00}_8$, b $= \mathbf{x02}_8$, p $= \mathbf{x80}_8$, and x $= \mathbf{x7e}_8$ the left-hand side evaluates to $\mathbf{x03}_8$ while the right-hand side evaluates to $\mathbf{xff}_8$. A counterexample is also found when the bit vector size is changed to 16.

(c) The replacement is correct since the following SMT encoding is unsatisfiable:

```
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const before (_ BitVec 8))
(declare-const after (_ BitVec 8))
(assert (= before (bvadd (bvsub #x00 a) (bvsub #x00 b))))
(assert (= after (bvsub #x00 (bvadd a b))))
(assert (not (= before after)))
(check-sat)
```

This is still true when changing the bit width to 16.

4 For $n = 2$, we obtain

```
(declare-const x (_ BitVec 32))
(declare-const y (_ BitVec 32))
(declare-const bit (_ BitVec 1))
(declare-const n (_ BitVec 32))
(assert (= y (bvand x (bvnot (bvshl #x00000001 #x00000002)))))
(assert (= bit ((_ extract 2 2) y)))
(assert (not (= bit #b0)))
(check-sat)
```