Automatic Abstraction Using Decision Procedures

Jörg Brauer
RWTH Aachen University
brauer@embedded.rwth-aachen.de

06.03.2012 @ Saarbrücken

Myself

- Diploma in CS from Kiel (2002-2008)
- 1,5 years at NICTA, Sydney (2006-2008)
  - Static analysis of C/C++
- Since 10/2008 at RWTH Aachen
  - Project leader for [mc]square since 01/2010, now ARCADE
  - Aalborg from 04/2011 till 06/2011
- Near future:
  - Submit dissertation in April
  - Work as "Verifikationsspezialist" with Verified Systems, Bremen

My Research Interests

- Automatic abstraction of binaries & decision procedures
  - with Andy King (Kent)
- Non-instrumenting runtime verification
  - with Thomas Reinbacher (TU Vienna)
- Model checking of PLCs
  - with Sebastian Biallas (RWTH)

Motivating Example

```
INC R0
MOV R1 R0
LSL R1
SBC R1 R1
EOR R0 R1
SUB R0 R1
```

- Goal: Affine transfer functions that relate interval boundaries
- Wraps are ubiquitous on 8-bit architecture
- Guard wrapping inputs using octagons [Min06]

Motivating Example

```
INC R0
MOV R1 R0
LSL R1
SBC R1 R1
EOR R0 R1
SUB R0 R1
```

- Block can over/underflow in three different ways

$$(\mathbf{r0} = 127)$$
$$(-128 \leq \mathbf{r0} \leq -2)$$
$$(-1 \leq \mathbf{r0} \leq 126)$$

Guards

Motivating Example

```
INC R0
MOV R1 R0
LSL R1
SBC R1 R1
EOR R0 R1
SUB R0 R1
```

- Block can over/underflow in three different ways

$$(\mathbf{r0} = 127) \Rightarrow (\mathbf{r0}' = -128)$$
$$(-128 \leq \mathbf{r0} \leq -2) \Rightarrow (\mathbf{r0}' = -\mathbf{r0} - 1)$$
$$(-1 \leq \mathbf{r0} \leq 126) \Rightarrow (\mathbf{r0}' = \mathbf{r0} + 1)$$

Guards (left-hand sides); Updates (right-hand sides)

Bit-Blasting

- Translate block into bit-vector logic, giving a formula φ_b [CKSY04]
- Enforce combination of wrapping behavior, e.g. `INC R0` and `SUB R0 R1` behave normally, but `LSL R2` overflows, denoted φ_w
- Then $\varphi = \varphi_b \wedge \varphi_w$ describes desired semantics
- Still need to abstract φ

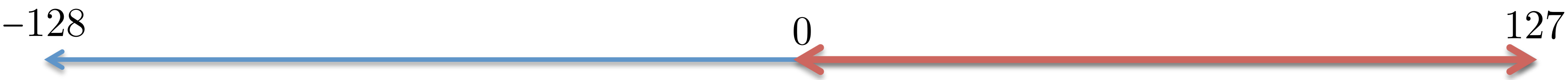
Abstracting φ With Linear Templates

- The block has input `R0` and output `R0`′
- Consider symbolic interval $\mathbf{r0} \in [\mathbf{r0}_l, \mathbf{r0}_u]$
- We know that $-128 \leq \mathbf{r0}_u \leq 127$
- Key idea: dichotomic search
- $\mathbf{r0}_u$ is uniquely determined, thus

$$(-128 \leq \mathbf{r0}_u \leq -1) \vee (0 \leq \mathbf{r0}_u \leq 127)$$

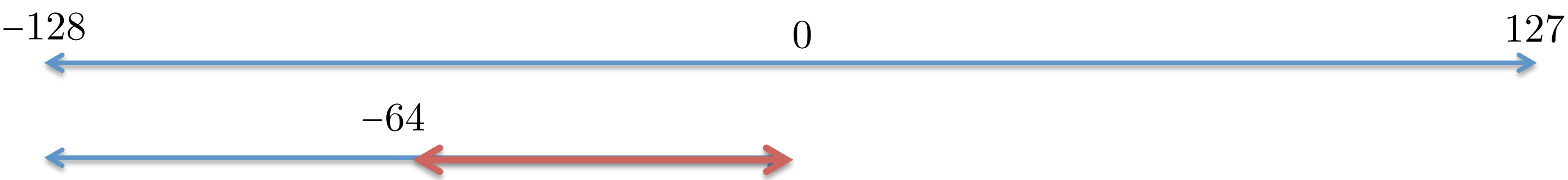
Abstracting φ Using Binary Search

Abstracting φ Using Binary Search



$$\varphi \wedge (\neg \mathbf{r0}[7]) \ ?$$

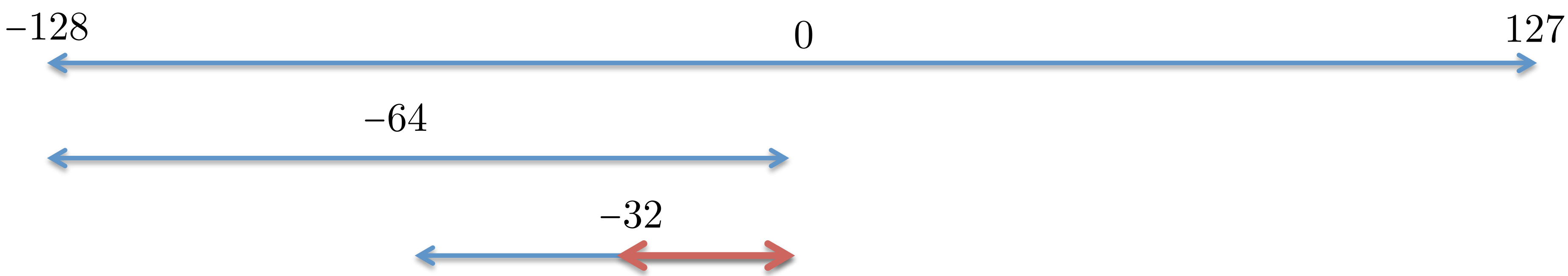
Abstracting φ Using Binary Search



$$\varphi \wedge (\neg \mathbf{r0}[7]) \ ? \ \text{no!}$$

$$\varphi \wedge (\neg \mathbf{r0}[7]) \wedge (\mathbf{r0}[6]) \ ?$$

Abstracting φ Using Binary Search

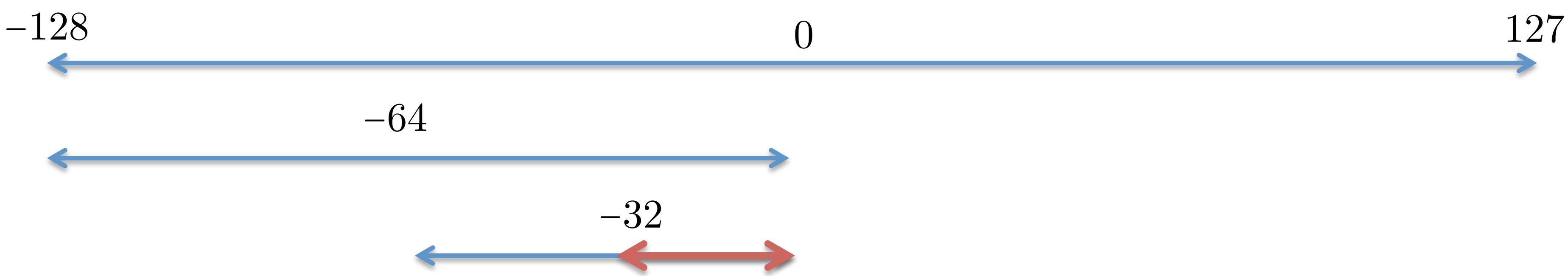


$$\varphi \wedge (\neg \mathbf{r0}[7]) \ ? \ \text{no!}$$

$$\varphi \wedge (\neg \mathbf{r0}[7]) \wedge (\mathbf{r0}[6]) \ ? \ \text{yes!}$$

$$\varphi \wedge (\neg \mathbf{r0}[7]) \wedge (\mathbf{r0}[6]) \wedge (\mathbf{r0}[5]) \ ?$$

Abstracting φ Using Binary Search



$$\varphi \wedge (\neg \mathbf{r0}[7]) \; ? \; \text{no!}$$

$$\varphi \wedge (\neg \mathbf{r0}[7]) \wedge (\mathbf{r0}[6]) \; ? \; \text{yes!}$$

$$\varphi \wedge (\neg \mathbf{r0}[7]) \wedge (\mathbf{r0}[6]) \wedge (\mathbf{r0}[5]) \; ? \; \text{yes!}$$

$$\Rightarrow \quad \mathbf{r0}_u = -2$$

Summary: Range Abstraction

- Characterize feasible inputs of some mode combination
- Simple form of dichotomic (or binary) search
- Efficient because of incremental SAT
- Can also be applied to octagons, etc.
- Alternative formulation using quantification is more complicated [Mon09,BK10]
  - Depends on alternating quantifiers, which is doable but difficult [BKK11]

Motivating Example Revisited

```
INC R0
MOV R1 R0
LSL R1
SBC R1 R1
EOR R0 R1
SUB R0 R1
```

- Block can over/underflow in three different ways

$$(\mathbf{r0} = 127) \Rightarrow (\mathbf{r0}' = -128)$$
$$(-128 \leq \mathbf{r0} \leq -2) \Rightarrow (\mathbf{r0}' = -\mathbf{r0} - 1)$$
$$(-1 \leq \mathbf{r0} \leq 126) \Rightarrow (\mathbf{r0}' = \mathbf{r0} + 1)$$

Done!

Next!

Abstracting φ With Affine Equalities

- Formula φ describes relation between $\mathbf{r0}$ on input and $\mathbf{r0}'$ on output
- Question: How to extract affine equality that relates $\mathbf{r0}$ and $\mathbf{r0}'$?
- Answer: Incremental affine hull [MS04]
  - Similar in in spirit to [RSY04]

Example: Affine Hull

- Pass φ to a SAT/SMT solver
- Obtain model $m_1 = (\mathbf{r0} = -4 \wedge \mathbf{r0}' = 3)$
- Represent as affine matrix

$$\left[\begin{array}{cc|c} 1 & 0 & -4 \\ 0 & 1 & 3 \end{array}\right]$$

Example: Affine Hull

- Current abstraction: $\left[\begin{array}{cc|c} 1 & 0 & -4 \\ 0 & 1 & 3 \end{array}\right]$ ←
- Pass $\varphi \wedge (\mathbf{r0}' \neq 3)$ to a SAT/SMT solver
- Obtain model $m_2 = (\mathbf{r0} = -5 \wedge \mathbf{r0}' = 4)$
- Represent as affine matrix and join

$$\left[\begin{array}{cc|c} 1 & 0 & -4 \\ 0 & 1 & 3 \end{array}\right] \sqcup \left[\begin{array}{cc|c} 1 & 0 & -5 \\ 0 & 1 & 4 \end{array}\right] = \left[\begin{array}{cc|c} 1 & 1 & -1 \end{array}\right]$$

Example: Affine Hull

- Current abstraction: $\left[\begin{array}{cc|c} 1 & 1 & -1 \end{array}\right]$ ←

- Pass $\varphi \wedge (\mathbf{r0} + \mathbf{r0}' \neq -1)$ to a SAT/SMT solver
- Formula is unsatisfiable
- Affine equality $\mathbf{r0}' = -\mathbf{r0} - 1$ thus over-approximates φ

Back to the Example

```
INC R0
MOV R1 R0
LSL R1
SBC R1 R1
EOR R0 R1
SUB R0 R1
```

- Block can over/underflow in three different ways

$$(\mathbf{r0} = 127) \Rightarrow (\mathbf{r0}' = -128)$$
$$(-128 \leq \mathbf{r0} \leq -2) \Rightarrow (\mathbf{r0}' = -\mathbf{r0} - 1)$$
$$(-1 \leq \mathbf{r0} \leq 126) \Rightarrow (\mathbf{r0}' = \mathbf{r0} + 1)$$

R0' := abs(R0 + 1)

Some More Properties

- Fairly quick: ~0.02s using Z3
- Can apply this technique to relate intervals and octagons, too [BK10,BK11]

$$
\begin{array}{lcl}
(\mathbf{r0}'_l = -128) & \wedge & (\mathbf{r0}'_u = -128) \\
(\mathbf{r0}'_l = -\mathbf{r0}_u - 1) & \wedge & (\mathbf{r0}'_u = -\mathbf{r0}_l - 1) \\
(\mathbf{r0}'_l = \mathbf{r0}_l + 1) & \wedge & (\mathbf{r0}'_u = \mathbf{r0}_u + 1)
\end{array}
$$

- Or even compute polynomial relations [MS04]

Experimental Results

Binary Program					$\overrightarrow{\mathcal{F}}$ interpreter			$\overrightarrow{\mathcal{F}} + \overleftarrow{\mathcal{B}}$ interpreter				
Name	Compiler	loc_C	$instr_B$	JT	RT	FT	Time	RS	k	RT	FT	Time
Single row input	KEIL SDCC	80	67 52	6	2401 460	2395 454	2.6 2.4	2 2	2 2	6 6	– –	3.32 2.0
Keypad	KEIL SDCC	113	113 80	9	3844 1508	3835 1499	3.49 3.08	4 4	2 2	9 9	– –	4.33 2.57
Communication Link	KEIL SDCC	111	164 118	8	6889 84	6881 76	4.56 3.38	2 2	2 2	8 8	– –	4.37 4.29
Task Scheduler	KEIL SDCC	81	105 97	5	>1000	>995	>5m	17 23	2 2	5 5	– –	14.03 10.23
Switch Case	KEIL SDCC	82	166 180	19	>5000 3304	>4981 3285	>5m 2.31	94 6	2 2	19 38	– 19	17.49 2.6
Emergency Stop	KEIL SDCC	138	150 141	9	768 256	759 247	2.8 2.9	2 2	2 2	9 9	– –	2.6 3.1

loc_C ... Lines of C code
$instr_B$ ... Number of assembly instructions
JT ... Number of jump targets
RT ... Number of recovered targets
FT ... Number of recovered false targets
RS ... Number of refinement steps applied
k ... Backtracking depth
Time ... Analysis time in seconds

So as to not cause Offense

- [RSY04] T. Reps, M. Sagiv and G. Yorsh: Symbolic Implementation of the Best Transformer (VMCAI'04)
- [RR04] J. Regehr and A. Reid: HOIST – A System for Automatically Deriving Static Analyzers for Embedded Systems (ASPLOS'04) [Mon09] D. Monniaux: Automatic Modular Abstractions for Linear Constraints (POPL'09)
- [TR12] A. Thakur and T. Reps: A Method for Symbolic Computation of Abstract Operations (TR-1708, University of Wisconsin)
- [Min06] A. Mine: The Octagon Abstract Domain (HOSC'06)
- [CKSY04] E. Clarke, D. Kroening, N. Sharygina and K. Yorav: Predicate Abstraction of ANSI-C Programs using SAT (FMSD'04)
- [MS04] M. Müller-Olm and H. Seidl: A Note on Karr's Algorithm (ICALP'04)

My Related Publications

- [BK10] J. Brauer and A. King: Automatic Abstraction for Intervals using Boolean Formulae (SAS'10)
- [BKK10] J. Brauer, A. King and S. Kowalewski: Range Analysis of Microcontroller Code using Bit-Level Congruences (FMICS'10)
- [BK11a] J. Brauer and A. King: Transfer Function Synthesis without Quantifier Elimination (ESOP'11 + hopefully LMCS'12)
- [BK11b] J. Brauer and A. King: Approximate Quantifier Elimination for Propositional Boolean Formulae (NFM'11)
- [BKK11] J. Brauer, A. King and J. Kriener: Existential Quantification as Incremental SAT (CAV'11)
- [RB11] T. Reinbacher and J. Brauer: Precise Control Flow Reconstruction using Boolean Logic (EMSOFT'11)
- [BKK12] J. Brauer, A. King and S. Kowalewski: Abstract Interpretation of Microcontroller Code: Intervals Meet Congruences (SCP'12)
- [BS12] J. Brauer and A. Simon: Inferring Counterexamples Through Under-Approximation (NFM'12)

Concluding Discussion

- We advocate automatic abstraction rather than manual design of transformers for binary analysis
  - Support for variety of different domains for free, e.g., affine equalities, intervals, value sets, octagons
  - Block-wise abstraction itself necessitates automation
- Key idea: decompose blocks based on wrapping modes
- SAT/SMT solvers can easily solve hundreds to thousands of instances per second now
- Future work: loop transformers + demand-driven abstraction

Thank you very much!