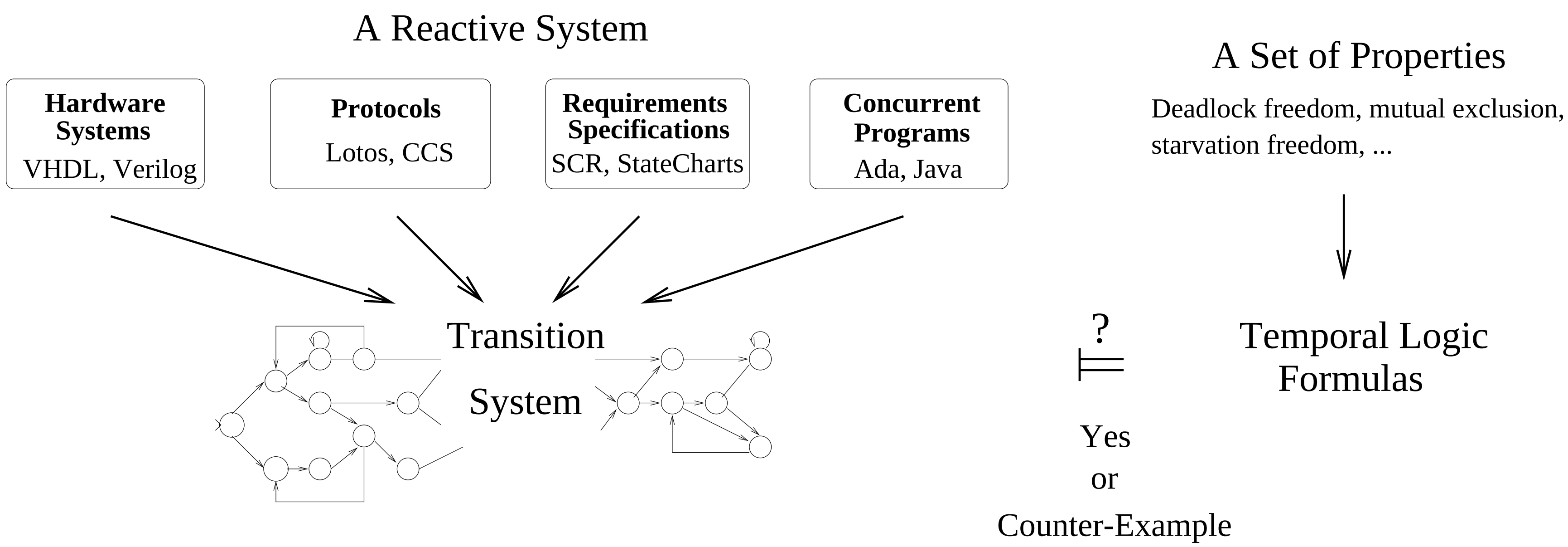
Automated Symbolic Analysis of Reactive Systems

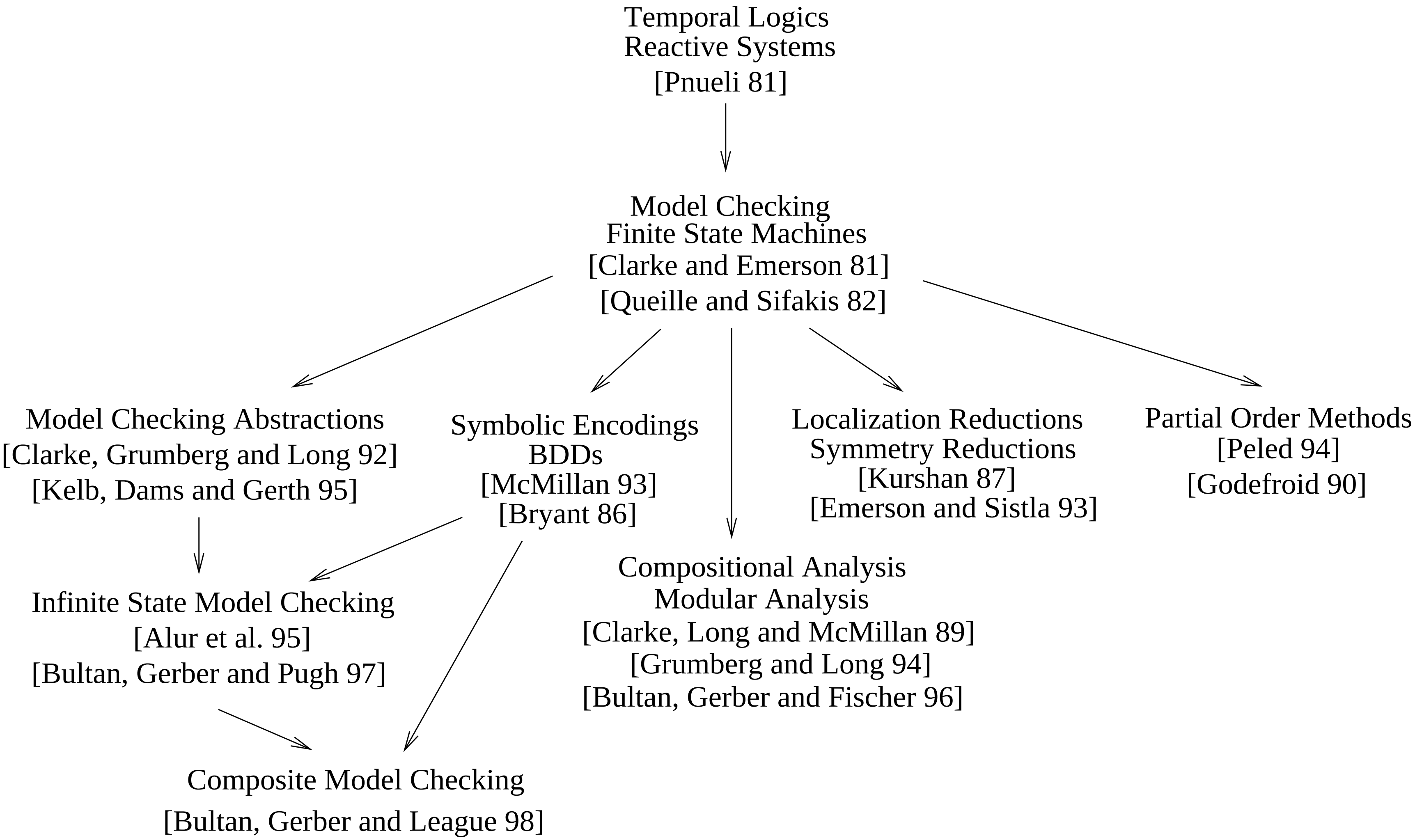
Tevfik Bultan

Department of Computer Science
University of Maryland, College Park

Model Checking Reactive Systems



Where Our Work Fits In



Our Research

- Handling unbounded integers in model checking
- Conservative approximation techniques
- Reachability analysis, partitioning techniques
- Handling integers and Booleans together efficiently

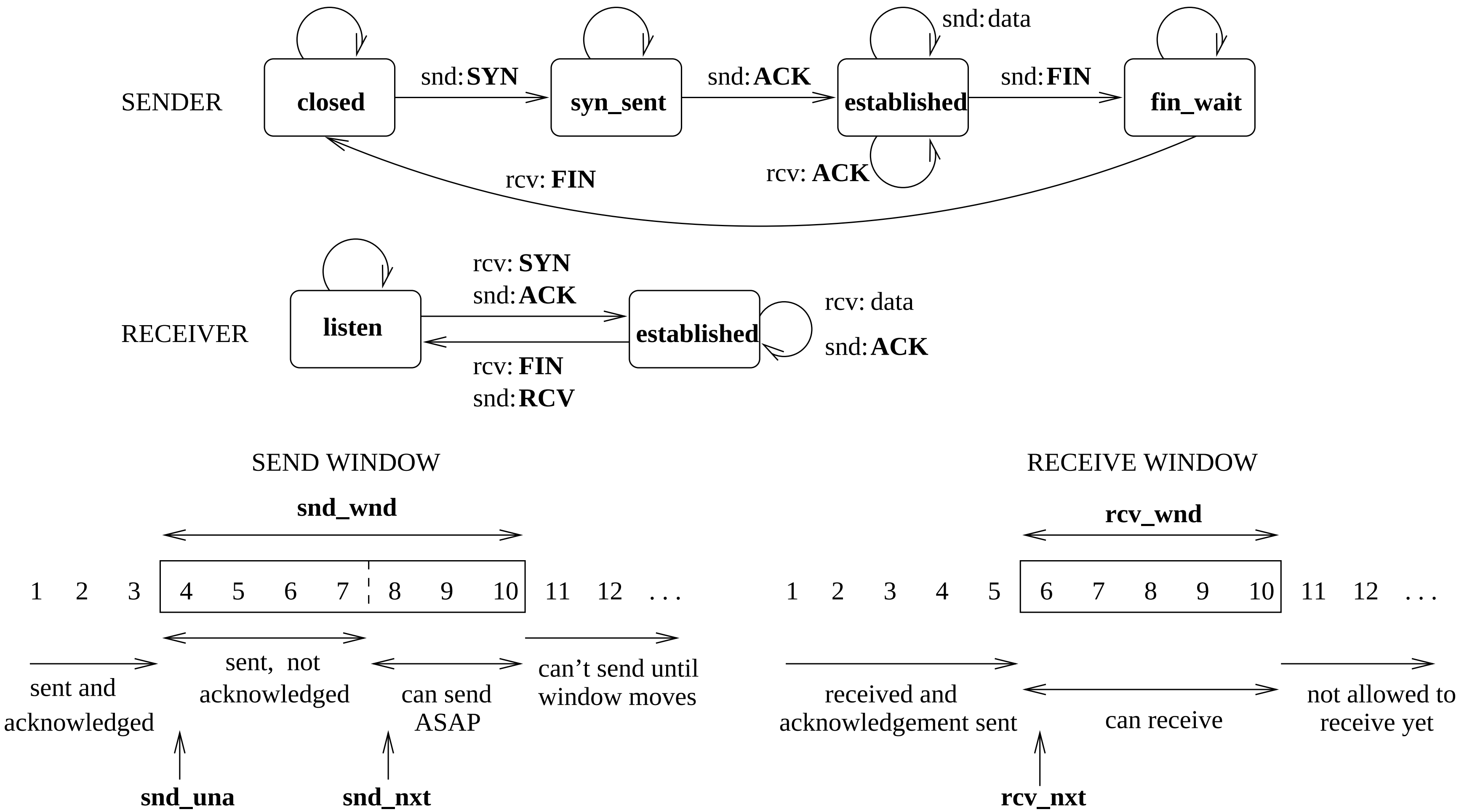
Problems which motivated these results:

- Mutual exclusion: Bakery algorithm, Ticket algorithm
- Bounded Buffers, Circular Queues, Producer/Consumer problems
- Protocols: TCP
- Requirements specifications: Safety Injection System

Outline

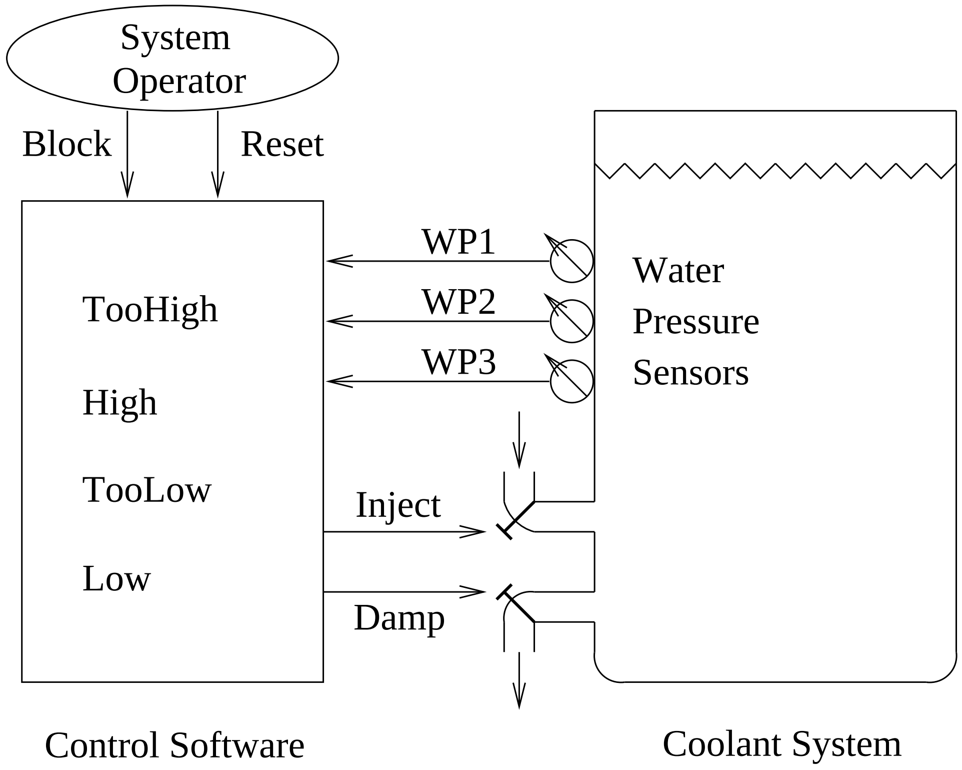
- Examples
- Prototype Analyzer:
  - Software architecture
  - Input programs and properties
- Symbolic Representations:
  - Booleans: BDDs
  - Integers: Presburger formulas
- Composite Model Checker
- Conservative Approximations
- Experimental Results
- Conclusions

Example 1: A Transport Protocol



Open Variables: **snd_wnd, rcv_wnd**

Example 2: Safety Injection System



- Original SCR specification [Courtois and Parnas 93]
- Finite state abstraction verified [Bharadwaj and Heitmeyer 97]

Safety Injection System: SCR Specification

Mode Class Pressure

Old Mode	Event	New Mode
–	@T(CTLow)	TooLow
–	@T(CLow)	Low
–	@T(CHigh)	High
–	@T(CTHigh)	TooHigh

Term Overridden

Mode	Events	
High	False	@T(InMode)
TooLow, Low, TooHigh	@T(Block=On) WHEN Reset=Off	@T(InMode) OR @T(Reset=On)
Overridden	True	False

Condition Inject

Mode	Conditions	
Low, High, TooHigh	True	False
TooLow	Overridden	NOT Overridden
Inject	Off	On

Condition Damp

Mode	Conditions	
TooLow, Low, High	True	False
TooHigh	Overridden	NOT Overridden
Damp	Off	On

$$\text{CTLow} \stackrel{\text{def}}{=} (\texttt{WP1, WP2} < \texttt{low}) \text{ OR } (\texttt{WP1, WP3} < \texttt{low}) \text{ OR } (\texttt{WP2, WP3} < \texttt{low})$$

$$\text{CLow} \stackrel{\text{def}}{=} (\texttt{low} \leq \texttt{WP1, WP2} < \texttt{high}) \text{ OR } (\texttt{low} \leq \texttt{WP1, WP3} < \texttt{high}) \text{ OR } (\texttt{low} \leq \texttt{WP2, WP3} < \texttt{high})$$

$$\text{CHigh} \stackrel{\text{def}}{=} (\texttt{high} \leq \texttt{WP1, WP2} < \texttt{toohigh}) \text{ OR } (\texttt{high} \leq \texttt{WP1, WP3} < \texttt{toohigh}) \text{ OR } (\texttt{high} \leq \texttt{WP2, WP3} < \texttt{toohigh})$$

$$\text{CTHigh} \stackrel{\text{def}}{=} (\texttt{toohigh} \leq \texttt{WP1, WP2}) \text{ OR } (\texttt{toohigh} \leq \texttt{WP1, WP3}) \text{ OR } (\texttt{toohigh} \leq \texttt{WP2, WP3})$$

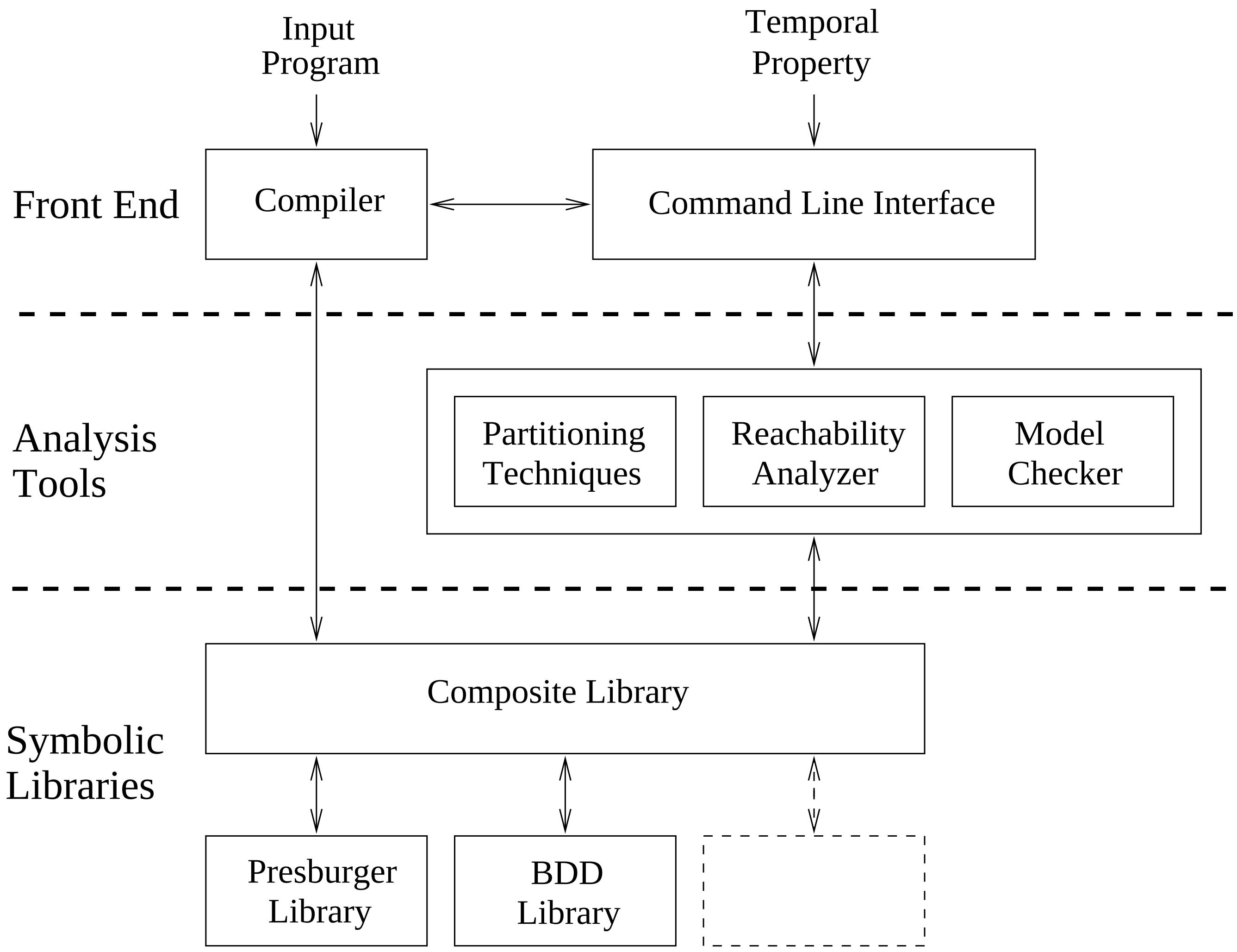
$$\texttt{min} \leq \texttt{WP1, WP2, WP3} \leq \texttt{max} \ \text{ AND } \ \texttt{min} < \texttt{low} < \texttt{high} < \texttt{toohigh} < \texttt{max}$$

Open Variables: `min`, `max`, `low`, `high`, `toohigh`

Outline

- Examples
- **Prototype Analyzer:**
  - Software architecture
  - Input programs and properties
- Symbolic Representations:
  - Booleans: BDDs
  - Integers: Presburger formulas
- Composite Model Checker
- Conservative Approximations
- Experimental Results
- Conclusions

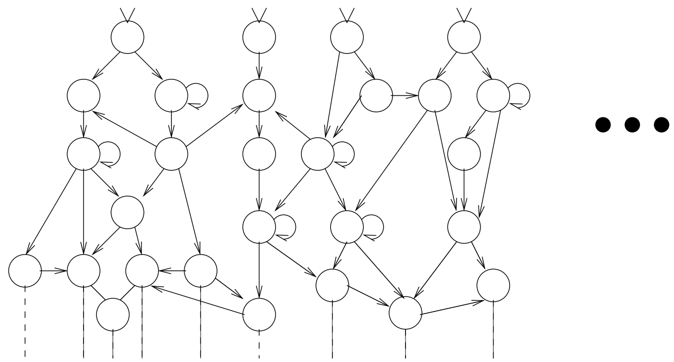
Prototype Analyzer: Software Architecture



Event-Action Language

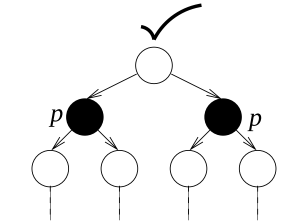
Input program	Semantics: A transition system
• A set of variables V	• Set of states $S = \mathbf{dom}(v_1) \times \mathbf{dom}(v_2) \ldots$
• An initial condition I	• Set of initial states I
• A set of atomic events E	• Transition relation $R \equiv \bigcup_{e \in E} R_e$

VARIABLES
$wp1, wp2, wp3$: integer
$Low, TLow$: boolean
INITIAL CONDITION
$Low \ \wedge \ \neg TLow \ \wedge \ min \leq wp1, wp2, wp3 \leq max$
EVENTS
e_1 : $(wp1', wp2' < low \ \vee \ wp1', wp3' < low \ \vee \ wp2', wp3' < low) \ \wedge \ TLow'$
e_2 : $(wp1, wp2 \geq low \ \vee \ wp1, wp3 \geq low \ \vee \ wp2', wp3' \geq low) \ \wedge \ Low'$
•
•
•

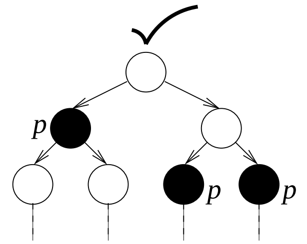


Basic Temporal Operators

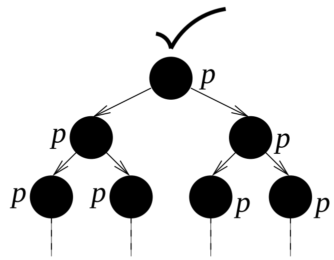
Operator	Meaning
AX p (EX p)	**All** paths satisfy p in the **next** state (**Some** paths)



Operator	Meaning
AF p (EF p)	**All** paths **eventually** satisfy p (**Some** paths)



Operator	Meaning
AG p (EG p)	**All** paths **invariantly** satisfy p (**Some** paths)



Example Properties

If the reset switch is on and the water pressure is too low, then safety injection should be active:

$$\mathsf{AG}(\ Reset\ \wedge\ TLow\ \implies\ Inject\ )$$

If two of the water pressure sensors show that water pressure is high, then the system is in either `High` or `TooHigh` mode:

$$\mathsf{AG}(\ high \leq wp1, wp2\ \implies\ (High\ \vee\ TooHigh)\ )$$

Model Checking: Goal

Given a program and a temporal property p :

- Either show that all initial states satisfy the temporal property p :

$$\text{set of initial states} \subseteq \text{truth set of } p$$

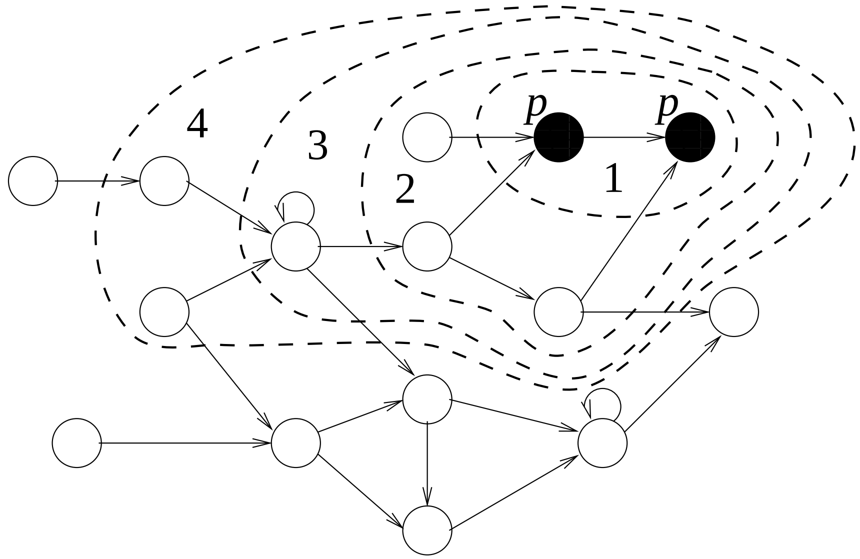
- Or find an initial state which does not satisfy the property p :

$$\text{a program state} \in \text{set of initial states} \cap \text{truth set of } \neg p$$

GOAL: Find a representation for the truth set of p which enables us to test these conditions

Temporal Properties $\equiv$ Fixpoints

$$\mathsf{EF}\ p \equiv \underbrace{\underbrace{\underbrace{\underbrace{p}_{1} \vee (\mathsf{EX}\ p)}_{2} \vee (\mathsf{EX}\ (\mathsf{EX}\ p))}_{3} \vee (\mathsf{EX}\ (\mathsf{EX}\ (\mathsf{EX}\ p)))}_{4} \vee \ \ldots$$



Tools Required for Model Checking

- Basic set operations for:
  $p \wedge q, \quad p \vee q, \quad \neg p$

```
Symbolic  And(Symbolic,Symbolic)
Symbolic  Or(Symbolic,Symbolic)
Symbolic  Not(Symbolic)
```

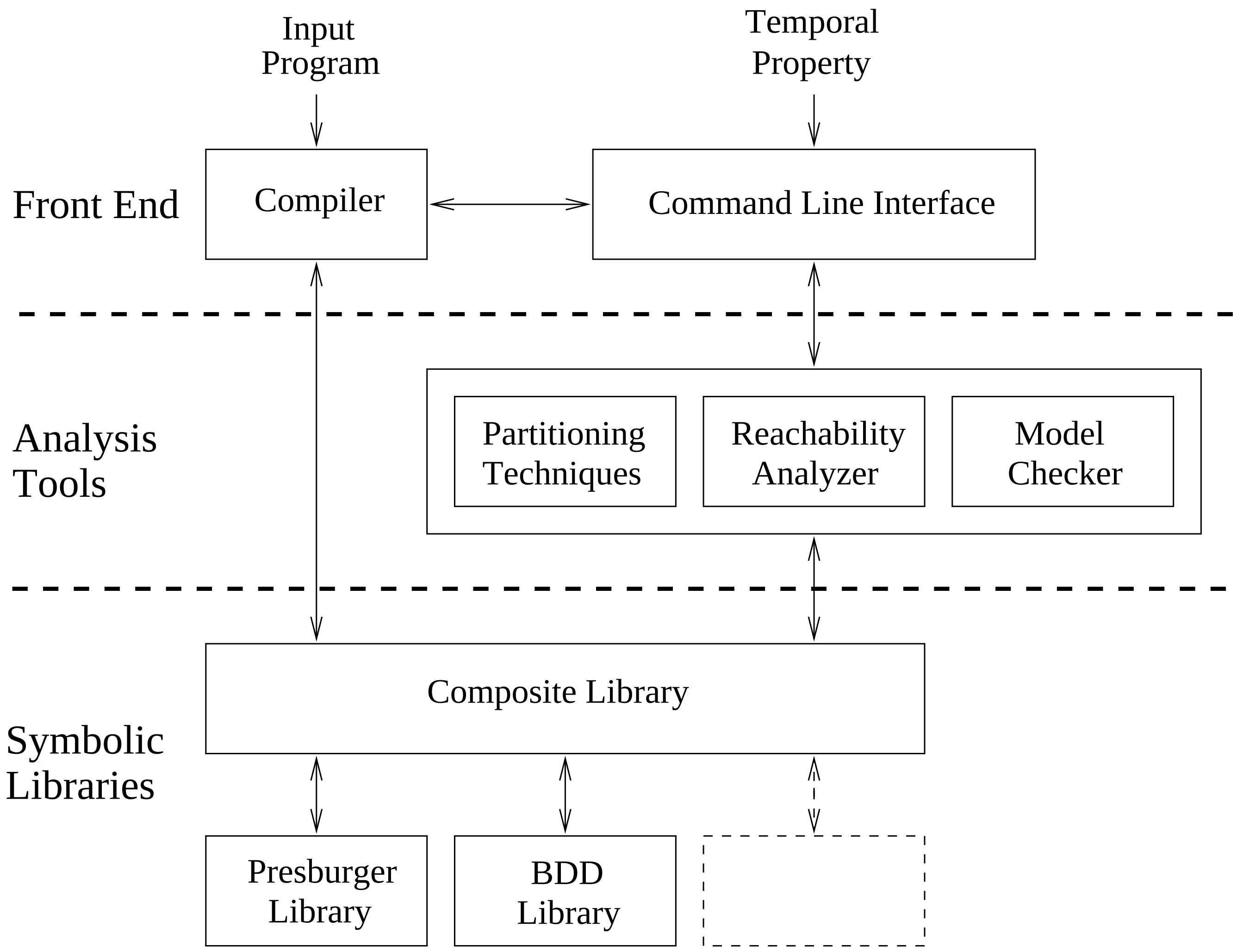
- Equivalence checking: $p \equiv q$?

```
Boolean  IsSatisfiable(Symbolic)
```

- Precondition function: EX p

```
Symbolic  EX(Symbolic)
```

Prototype Analyzer: Software Architecture

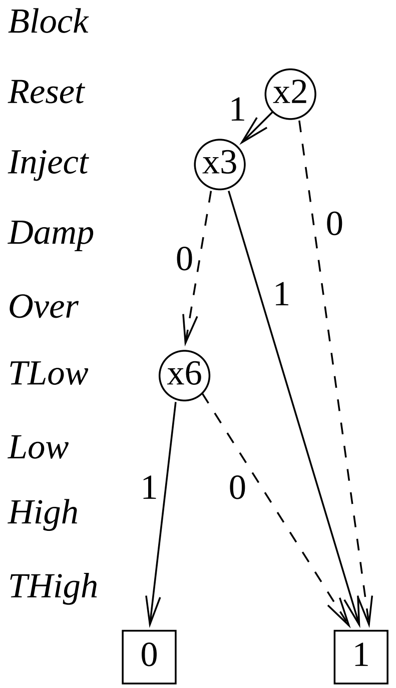


Outline

- Examples
- Prototype Analyzer:
  - Software architecture
  - Input programs and properties
- **Symbolic Representations:**
  - Booleans: BDDs
  - Integers: Presburger formulas
- Composite Model Checker
- Conservative Approximations
- Experimental Results
- Conclusions

Representing Boolean States: BDDs

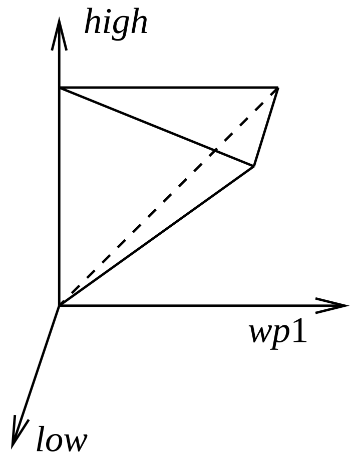
- How can we represent a Boolean set q?
- Variables: $Block, Reset, Inject, Damp, Over, TLow, Low, High, THigh$

Explicit State Enumeration	Boolean Logic	BDD
$\{FFFFFFFFF, FFFFFFFFT,$ $FFFFFFFTF, FFFFFFFTT,$ $FFFFFFTFF, FFFFFFTFT,$ $FFFFFFTTF, \ldots\}$	$Reset \wedge TLow \implies Inject$ $\equiv \neg Reset \vee \neg TLow \vee Inject$	

Representing Integer States: Affine Constraints

- How can we represent an integer set q?
- Variables: $wp1, wp2, wp3, min, max, low, high, toohigh$

Explicit State Enumeration ?	Affine Constraints
$\{(0,0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,1,0),$ $(1,0,0,0,0,0,0,1,0),(0,0,0,0,0,0,0,2,0),$ $(1,0,0,0,0,0,1,1,0),\ldots$	$0 \leq low \leq wp1 \leq high$



Presburger Formula Representation

Represent integer sets using Presburger formulas:

- Operator $+$, multiplication by a constant
- Predicates $\leq$, $\geq$, $=$, $<$, $>$
- Quantification $\exists$, $\forall$

All the functionality required for model checking can be supported for this representation (e.g., Omega library).

Problems with Presburger Representation

Problem 1 It is not efficient to encode Boolean variables as integers

Problem 2 Model checking is undecidable for infinite domains

Our solutions:

Solution 1 Use a composite representation

- Encode Booleans with BDDs
- Encode integers with Presburger formulas

Solution 2 Use conservative approximation techniques

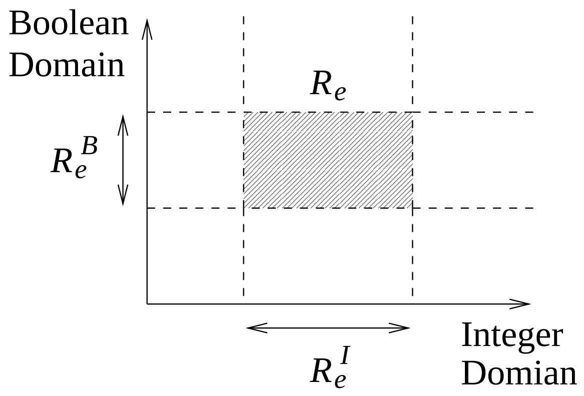
Outline

- Examples
- Prototype Analyzer:
  - Software architecture
  - Input programs and properties
- Symbolic Representations:
  - Booleans: BDDs
  - Integers: Presburger formulas
- **Composite Model Checker**
- Conservative Approximations
- Experimental Results
- Conclusions

Solution 1: Composite Representations

Conjunctively partition each event e:

$$R_e \equiv R_e^B \ \wedge \ R_e^I$$



R_e^B: BDD encoding $\qquad$ R_e^I: Presburger encoding

Sets of states:

$$q \equiv \bigvee_i \ (q_i^B \ \wedge \ q_i^I)$$

q_i^B: BDD encoding $\qquad$ q_i^I: Presburger encoding

Composite Relational Image Computations

$$\mathsf{EX}\ q \stackrel{\mathrm{def}}{=} \{\ s \mid \exists s'\ .\ s' \in q\ \wedge\ (s, s') \in R\ \}$$

$$\begin{aligned}
\mathsf{EX}\ q \quad &\equiv \quad R^{-1}\ (\ \bigvee_i q_i\ ) \\
&\equiv \quad \bigvee_{e \in E}\ \bigvee_i\ R_{\mathbf{e}}^{-1}\ (\ q_i\ ) \\
&\equiv \quad \bigvee_{e \in E}\ \bigvee_i\ R_{\mathbf{e}}^{-1}\ (\ q_i^B\ \wedge\ q_i^I\ ) \\
&\equiv \quad \bigvee_{e \in E}\ \bigvee_i\ (R_{\mathbf{e}}^{\mathbf{B}})^{-1}(q_i^B)\ \wedge\ (R_{\mathbf{e}}^{\mathbf{I}})^{-1}(q_i^I)
\end{aligned}$$

Composite Model Checker

```
Procedure Eval(f)

  Case

    IsEvaluated(f)   :   Return(f)

    f = ¬f1          :   Return(Not(Eval(f1)))
    f = f1 ∧ f2      :   Return(And(Eval(f1),Eval(f2)))
    f = f1 ∨ f2      :   Return(Or(Eval(f1),Eval(f2)))

    f = EX f1        :   Return(EX(Eval(f1)))
    f = AX f1        :   Return(Not(EX(Not(Eval(f1)))))

    f = f1 EU f2     :   q0 ≡ f2    q(i+1) ≡ Eval(qi ∨ (f1 ∧ EX qi))
                         Return(qn) when qn ≡ q(n+1)

    f = f1 AU f2     :   q0 ≡ f2    q(i+1) ≡ Eval(qi ∨ (f1 ∧ EX qi ∧ AX qi))
                         Return(qn) when qn ≡ q(n+1)
```

In mathematical notation:

- $\mathtt{IsEvaluated}(f)$: $\mathtt{Return}(f)$
- $f = \neg f_1$: $\mathtt{Return(Not(Eval(}f_1\mathtt{)))}$
- $f = f_1 \wedge f_2$: $\mathtt{Return(And(Eval(}f_1\mathtt{),Eval(}f_2\mathtt{)))}$
- $f = f_1 \vee f_2$: $\mathtt{Return(Or(Eval(}f_1\mathtt{),Eval(}f_2\mathtt{)))}$
- $f = \mathsf{EX}\ f_1$: $\mathtt{Return(EX(Eval(}f_1\mathtt{)))}$
- $f = \mathsf{AX}\ f_1$: $\mathtt{Return(Not(EX(Not(Eval(}f_1\mathtt{)))))}$
- $f = f_1\ \mathsf{EU}\ f_2$: $q_0 \equiv f_2 \quad q_{i+1} \equiv \mathtt{Eval}(q_i \vee (f_1 \wedge \mathsf{EX}\ q_i))$; $\mathtt{Return}(q_n)$ when $q_n \equiv q_{n+1}$
- $f = f_1\ \mathsf{AU}\ f_2$: $q_0 \equiv f_2 \quad q_{i+1} \equiv \mathtt{Eval}(q_i \vee (f_1 \wedge \mathsf{EX}\ q_i \wedge \mathsf{AX}\ q_i))$; $\mathtt{Return}(q_n)$ when $q_n \equiv q_{n+1}$

Outline

- Examples
- Prototype Analyzer:
  - Software architecture
  - Input programs and properties
- Symbolic Representations:
  - Booleans: BDDs
  - Integers: Presburger formulas
- Composite Model Checker
- **Conservative Approximations**
- Experimental Results
- Conclusions

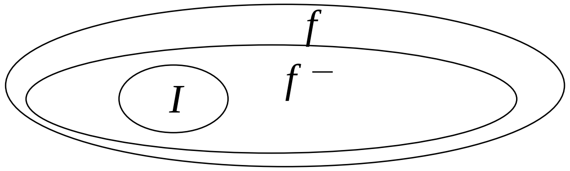
Solution 2: Conservative Analysis

- Given a temporal formula f, what we are really interested in is:
  - Either to show that $I \subseteq f$
  - Or to find a counter-example $s \in I \cap \neg f$

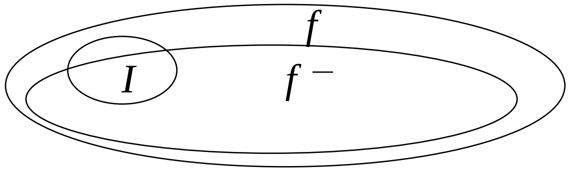
- **Problem:** Computing truth set of f is undecidable

- Compute a **lower bound** f^-, $f^- \subseteq f$

  If $I \subseteq f^-$ then $I \subseteq f$ hence, f holds



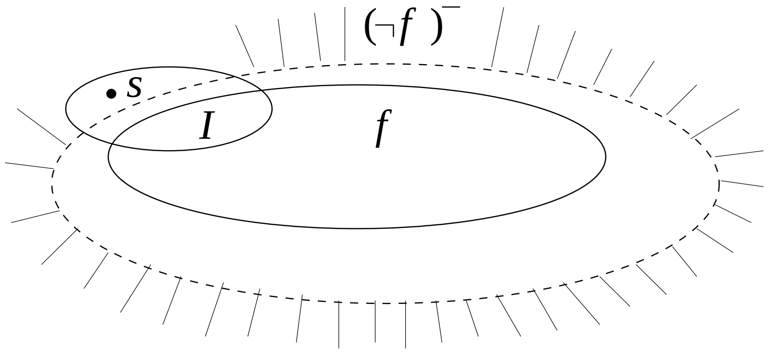
- However, if $I \not\subseteq f^-$, we can conclude nothing.



Conservative Analysis

- Find a lower bound for $\neg f$, i.e., $(\neg f)^-$
- If we find some s such that

  $$s \in I \cap (\neg f)^- \quad \text{then} \quad I \not\subseteq f$$



  We generate a counter-example starting from s.

- If both

  $$I \not\subseteq f^- \quad \text{and} \quad I \cap (\neg f)^- \equiv \emptyset$$

  then analysis is inconclusive.

Computing Lower Bounds

- Model checking is recursive: subformulas are evaluated first
  - If top level operator is not negation:

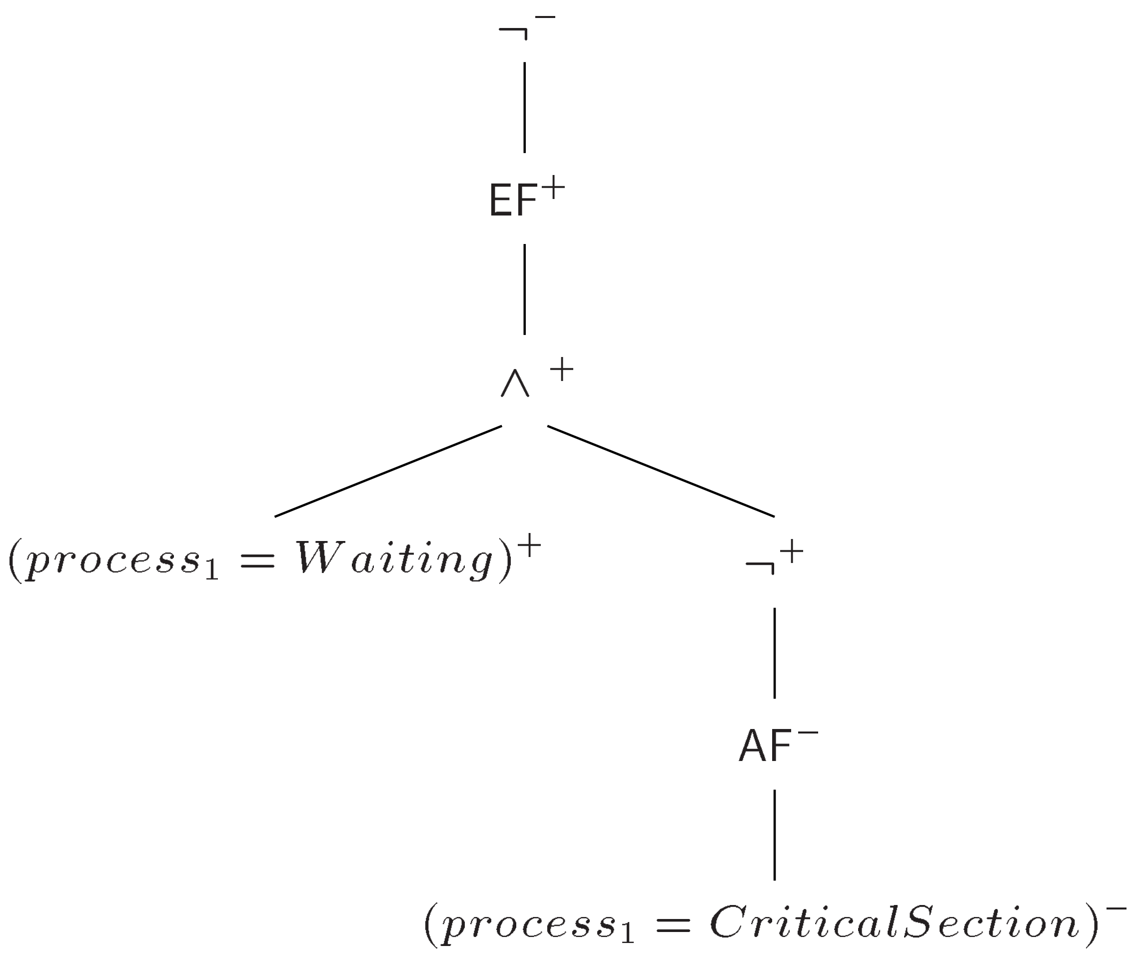
    To compute a lower bound for a formula:
    $\rightarrow$ Compute lower bounds for subformulas

  - Negation operator?

    $\rightarrow$ Compute an upper bound for the subformula
    $\rightarrow (\neg f)^- \equiv (\neg f^+) \equiv S - f^+$

Labeling Algorithm

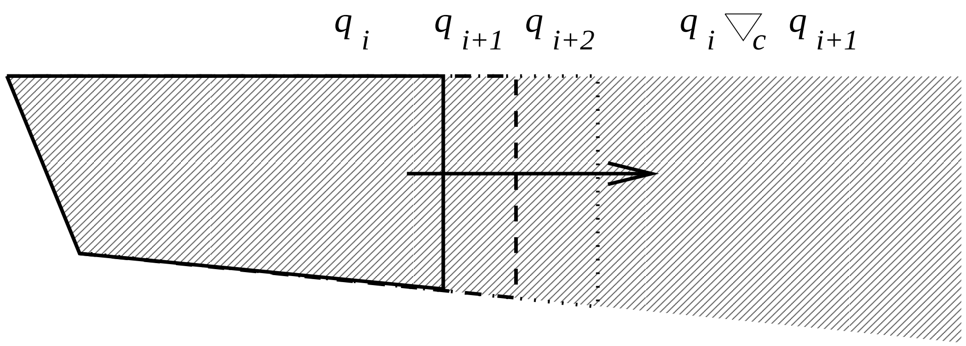
$$\neg(\mathsf{EF}(process_1 = Waiting \ \wedge \ \neg\mathsf{AF}(process_1 = CriticalSection)))$$



Computing Upper Bounds

- Given a fixpoint computation $q_0, q_1, q_2, \ldots$ s.t. $q_i \subseteq q_{i+1}$
- Expand (widen) the iterates in the direction of growth
- A widening operator ($\bigtriangledown_c$) for convex sets:

  $q_i \ \bigtriangledown_c \ q_{i+1} \equiv$ Conjunction of constraints of q_i which are also satisfied by q_{i+1}



Multi-Polyhedra Widening

$$q_i \equiv p_1 \lor p_2 \lor \ldots \lor p_m$$
$$q_{i+1} \equiv r_1 \lor r_2 \lor \ldots \lor r_m \lor r_{m+1} \lor \ldots \lor r_n$$

Constraints: $m \leq n$, and for all $1 \leq i \leq m,\ p_i \subseteq r_i$

$$q_i \quad \nabla \quad q_{i+1}$$
$$\equiv \quad (p_1 \lor p_2 \lor \ldots \lor p_m) \quad \nabla \quad (r_1 \lor r_2 \lor \ldots \lor r_m \lor r_{m+1} \lor \ldots \lor r_n)$$
$$\equiv \quad p_1 \nabla_c r_1 \lor p_2 \nabla_c r_2 \lor \quad \ldots \quad \lor p_m \nabla_c r_m \lor r_{m+1} \lor \ldots \lor r_n$$

Computing Lower Bounds

- Any iterate in the exact fixpoint computation sequence gives a lower bound

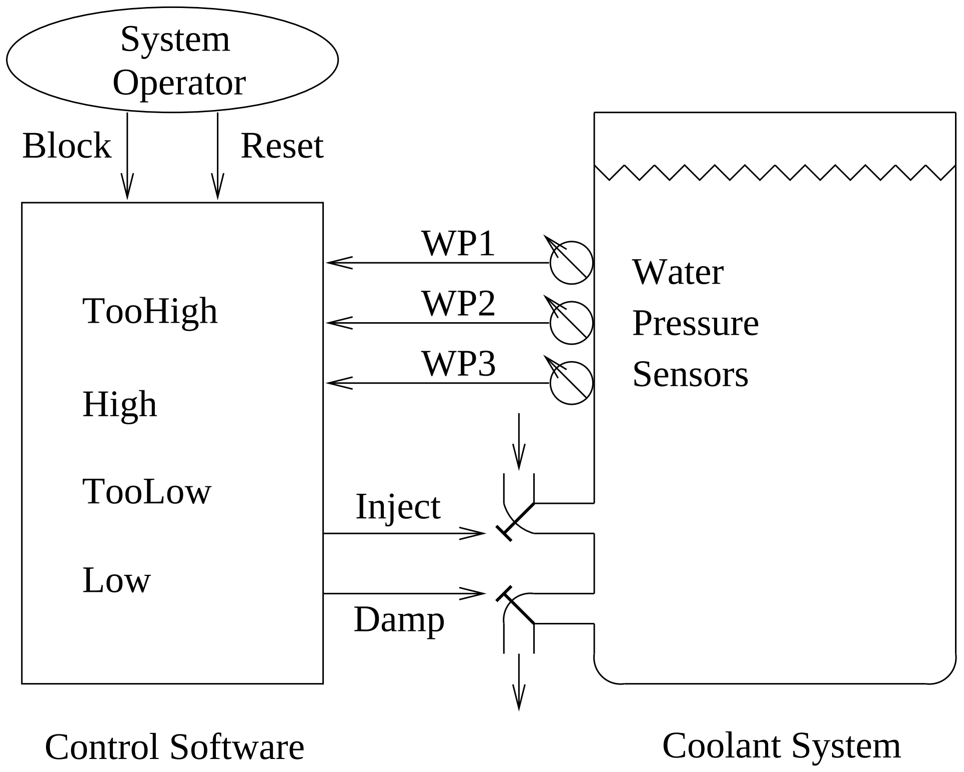
- Stop the fixpoint iterations using a time limit to get a lower bound

- Store partial results and improve if necessary

Outline

- Examples
- Prototype Analyzer:
  - Software architecture
  - Input programs and properties
- Symbolic Representations:
  - Booleans: BDDs
  - Integers: Presburger formulas
- Composite Model Checker
- Conservative Approximations
- **Experimental Results**
- Conclusions

Safety Injection System



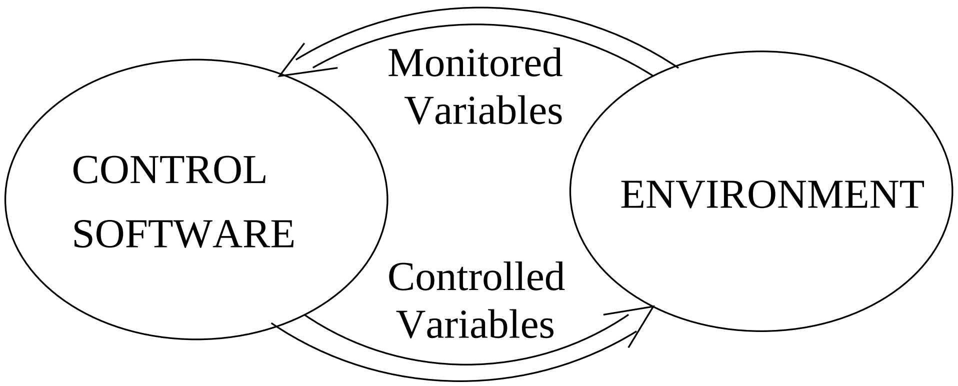
Experiments: Safety Injection System, Safety

PROPERTY	EXEC. TIME
(P1) AG$((Reset \wedge TLow) \implies Inject))$	2.58 sec.
(P2) AG$(Inject \implies TLow)$	1.52 sec.
(P3) AG$((low \leq wp1, wp2 < high \vee low \leq wp1, wp3 < high \vee$ $low \leq wp2, wp3 < high) \iff Low)$	35.54 sec.
(P4) AG$((toohigh \leq wp1, wp2 \vee toohigh \leq wp1, wp3 \vee$ $toohigh \leq wp2, wp3) \iff THigh)$	15.94 sec.
(P5) AG$((high \leq wp1, wp2) \implies (High \vee TooHigh))$	98.51 sec.

Experiments: Safety Injection System, Liveness

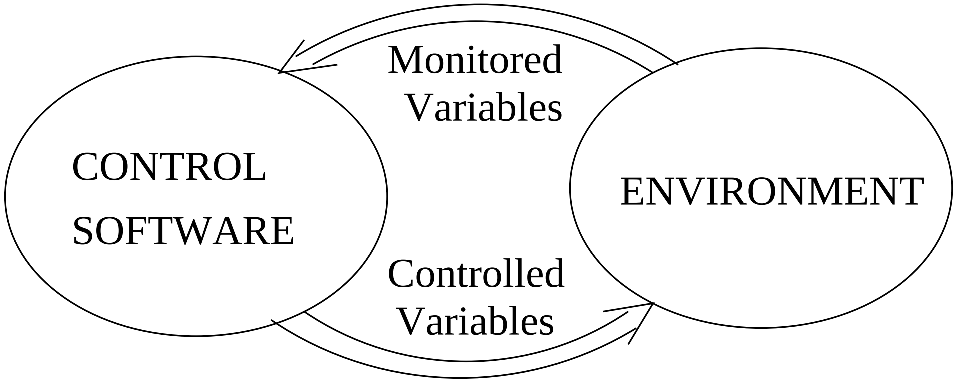
If the safety injection stays active, then eventually the water pressure will be OK:

$$\mathsf{AG}(\ (Inject)\ \mathsf{AU}\ (low \leq wp1)\ )$$



We have to close the feedback loop!

Experiments: Safety Injection System, Liveness

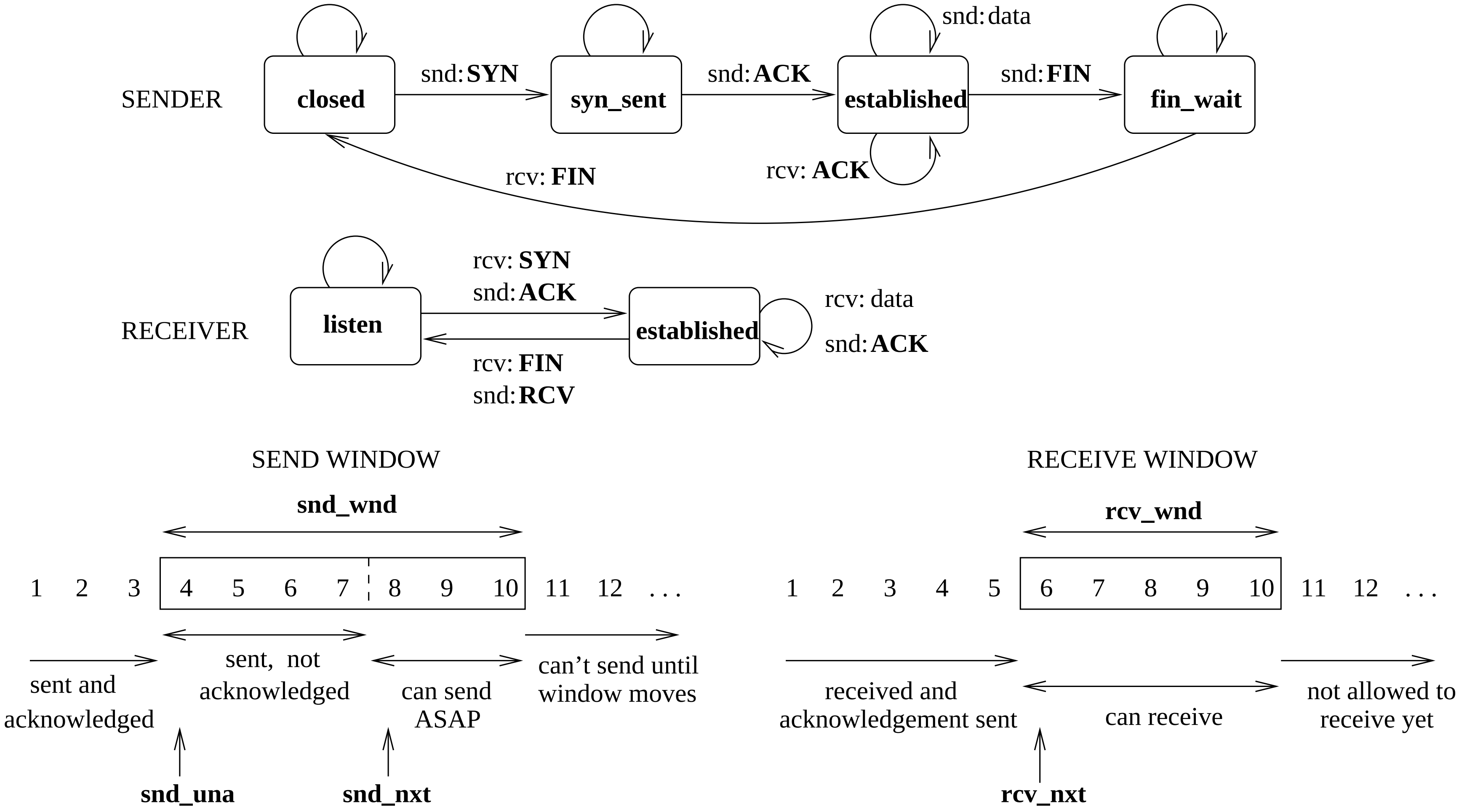


Environmental hypothesis:

$$Inject \;\wedge\; wp1 = i \quad\Longrightarrow\quad \mathsf{AX}(\; wp1 > i + (low - min)\;)$$

Using this hypothesis the liveness property is verified in 32.11 seconds.

Transport Protocol



Experiments: Transport Protocol

PROPERTY	FIXPOINT	CONVERGED IN
(P1) $\mathsf{AG}(sender = \texttt{established} \implies receiver = \texttt{established})$	Exact	4 iter. – 3.11 sec.
	Approx.	4 iter. – 3.42 sec.
(P2) $\mathsf{AG}(receiver = \texttt{established} \implies \neg(sender = \texttt{closed}))$	Exact	did not converge
	Approx.	6 iter. – 14.13 sec.
(P3) $\mathsf{AG}(sender = \texttt{established} \implies snd\_una \leq snd\_nxt \leq snd\_una + snd\_wnd)$	Exact	1 iter. – 1.74 sec.
	Approx.	1 iter. – 1.93 sec.
(P4) $\mathsf{AG}((receiver = \texttt{established} \wedge rcv\_nxt = i) \implies \mathsf{AX}(rcv\_nxt \leq i + rcv\_wnd))$	Exact	1 iter. – 1.73 sec.
	Approx.	1 iter. – 1.71 sec.
(P5) $\mathsf{AG}(sender = receiver = \texttt{established} \implies snd\_una \leq rcv\_nxt)$	Exact	7 iter. – 91.97 sec.
	Approx.	4 iter. – 11.52 sec.
(P6) $\mathsf{AG}(sender = receiver = \texttt{established} \implies rcv\_nxt \leq snd\_nxt)$	Exact	did not converge
	Approx.	10 iter. – 377.44 sec.

Summary of Contributions

- Presburger model checking for analyzing systems with integer variables
- Composite model checking using type-specific symbolic encodings
- Conservative approximation techniques
- Reachability analysis, partitioning heuristics

Current and Future Work

- Extending the composite library to other types (queues, reals, etc.)

- Conservative approximations: A framework for compositional and incremental analysis

- Analyzing more, bigger problems

Other Analysis Techniques: Partitioning

- Partition the state space: $P = \{S_1, S_2, \ldots, S_n\}$

  Represent a state set $q \subseteq S$ using one composite formula per partition class

$$q \equiv \bigvee_{S_i \in P} q \wedge S_i$$

$$\begin{aligned}
\mathsf{EX}\ q &\equiv R^{-1}\ (\bigvee_{S_i \in P} q \wedge S_i) \\
&\equiv \bigvee_{S_i \in P} (R^{-1}\ \ (q \wedge S_i)) \\
&\equiv \bigvee_{S_i \in P} \bigvee_{e \in E} (R_e^{-1}\ \ (q \wedge S_i))
\end{aligned}$$

Other Analysis Techniques: Reachability

- Use fixpoint approximation techniques to compute an upper bound RS^+ to the set of reachable states RS

  Restrict the fixpoint iterates to RS^+ to increase the convergence rate.

- Given two transition relations R, R', such that $R \subseteq R' \subseteq R^*$ the *reachability properties* of R and R' are the same.

  Compute EF p, AG p, and RS using R' to increase the convergence rate.

More Experiments

Ticket mutual exclusion algorithm:

$$\mathsf{AG}(\neg(process_1 = CriticalSection \ \wedge \ process_2 = CriticalSection)) \ \ 7.32 \text{ seconds}$$
$$\mathsf{AG}(process_1 = Waiting \implies \mathsf{AF}(process_1 = CriticalSection)) \ \ 27.03 \text{ sec.}$$

A bounded buffer with two producers and two consumers:

$$\mathsf{AG}(0 \leq p_1 + p_2 - (c_1 + c_2) \leq size) \ \ 5.34 \text{ sec.}$$

A circular queue:

$$\mathsf{AG}(head \leq size \ \wedge \ tail \leq size) \ \ 0.54 \text{ sec.}$$
$$\mathsf{AG}(tail \geq head \implies p - c = tail - head) \ \ 32.89 \text{ sec.}$$
$$\mathsf{AG}(tail < head \implies p - c = size - (head - tail) + 1) \ \ 37.06 \text{ sec.}$$
$$\mathsf{AG}(0 \leq p - c \leq size) \ \ 119.44 \text{ sec.}$$