# Easy Parameterized Verification of Biphase Mark and 8N1 Protocols

Geoffrey M. Brown, Indiana University
geobrown@cs.indiana.edu

Lee Pike (Presenting), Galois Connections[1]
leepike@galois.com

March 27, 2006

[1] Some of this work was performed at the NASA Langley Research Center

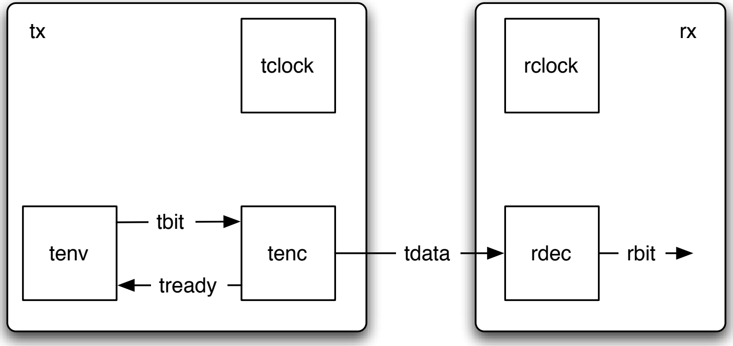
# Application: Biphase Mark and 8N1 Protocols

- Biphase Mark Protocol (BMP)

  Used for data transmission in CDs and ethernet, for example.
- 8N1 Protocol

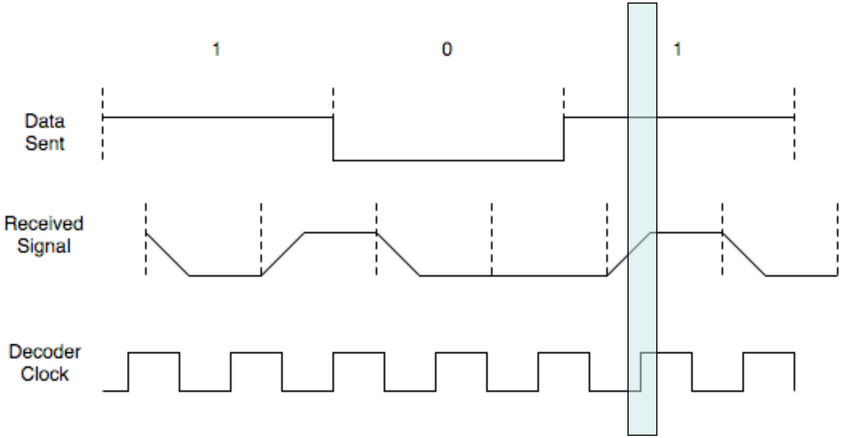
  Used in and UARTs.

# General System Architecture




Just the encoder (`tenc`), decoder (`rdec`), and constraints are protocol-specific.

# Unreliable Sampling



# What Makes This Hard?

- We're crossing clock domains
    - . . . With different phases, frequencies, and settling times and stable periods
    - . . . And error in these parameters due to jitter, signal skew, distortion, etc.
- And we want a *parameterized* verification
- So we want to prove correct behavior under general constrains on the parameters

# An Informal Comparison

- **Previous Efforts**:
  - *Mechanical theorem-proving*:
    - Using PVS (twice)
    - Using (a precursor to) ACL2
  - *Real-time model checking*:
    - HyTech
    - Uppaal
- **Our Effort**: The SAL infinite-state bounded model-checker combines SAT-solving and SMT decision procedures to *prove* safety properties about infinite-state models.

# An Informal Comparison

- One PVS effort required 37 invariants and 4000 individual proof directives (before "optimizing" the proofs).
- Ours required five invariants, each of which is proved automatically by SAL.

- In the other PVS effort, it takes 5 hours for PVS to *check* the manually-generated proof scripts.
- Ours requires just a few minutes to *generate* the proofs.

# An Informal Comparison

- The PVS efforts likely required months to complete. Presumably due to the difficulty of the endeavor, J. Moore reports the BMP verification as one of his "best ideas" in his career on his webpage.
- Our initial effort in SAL took *a couple days*.
  ...and we found a significant bug in a UART application note.

# What's Needed for Easy Parameterized Verification?

- A new automated proof technique
  (induction via infinite-state bounded model-checking)
- Expressive modeling language (SAL)
- Easy generation of invariants
  - $k$-induction
  - Disjunctive invariants

# SAL's Language

- Typed with predicate subtypes.
- Infinite types (e.g., `INTEGER` and `REAL`).
- Synchronous (lock-step) and asynchronous (interleaving) composition (`||` and `[]`, respectively).
- Quantification (over finite types).
- Recursion (over finite types).

# Easy Invariant Construction

*k-Induction* to strengthen invariants *automatically*.

- Generalizes induction over transition systems.
- Automatic but exponential in the size of the size of $k$.

# Induction (over Transition Systems)

Let $\langle S, S^0, \rightarrow \rangle$ be a transition system.

For safety property $P$, show

- **Base**: If $s \in S^0$, then $P(s)$;
- **Induction Step**: If $P(s)$ and $s \rightarrow s'$, then $P(s')$.

Conclude that for all reachable $s$, $P(s)$.

# $k$-Induction Generalization

Generalize from single transitions to trajectories of fixed length.

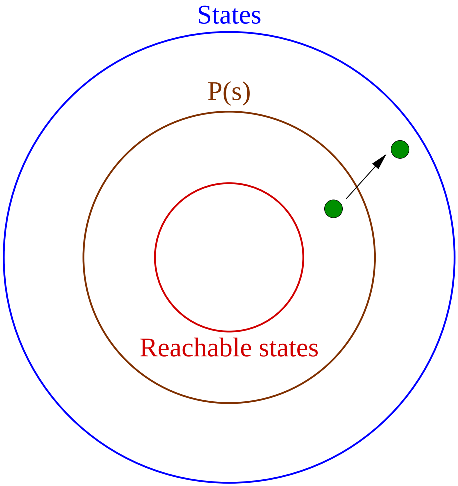
For safety property $P$, show

- **Base**: If $s_0 \in S^0$, then for all trajectories $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_k$, $P(s_i)$ for $0 \leq i \leq k$;
- **IS**: For all trajectories $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_k$, If $P(s_i)$ for $0 \leq i \leq k-1$, then $P(s_k)$.

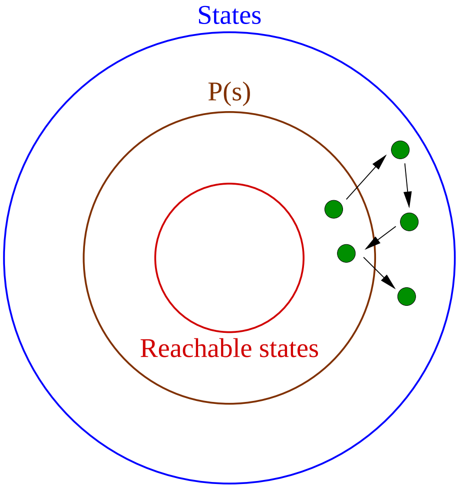
Conclude that for all reachable $s$, $P(s)$.

Induction is the special case when $k = 1$.

# Induction



# *k*-Induction



# *k*-Induction

```
counter1: MODULE =
  BEGIN
    LOCAL cnt : INTEGER
    LOCAL b   : BOOLEAN
    INITIALIZATION
      cnt = 0;
      b = TRUE
    TRANSITION
      [       b --> cnt' = cnt + 2;
                    b' = NOT b
        [] ELSE --> cnt' = cnt - 1;
                    b' = NOT b
      ] END;

 Thm1 : THEOREM counter1 |- G(cnt >= 0);
```

Circuit behavior:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| b = | T | F | T | F | T | F | ... |
| cnt = | 0 | 2 | 1 | 3 | 2 | 4 | ... |

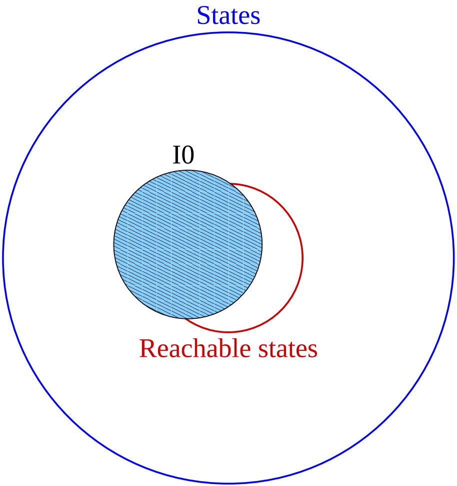
Thm1 fails for $k = 1$, succeeds for $k = 2$ (why?).

# Disjunctive Invariants

*Disjunctive Invariants* to weaken safety properties until they become invariant.

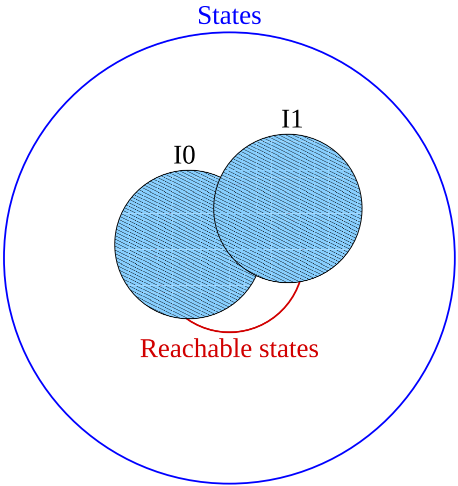
- General and interactive.
- Developed by Pneuli & Rushby, independently.
- A disjunctive invariant can be built iteratively to cover the reachable states from the counterexamples returned by SAL for the hypothesized invariant being verified.

# Initial Attempt



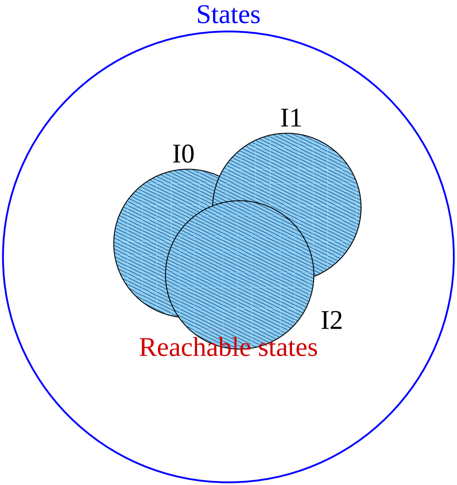

$I0$ Not invariant...

# Refinement




$I0 \vee I1$ Almost...

# Invariant




$I0 \vee I1 \vee I2$ There we go!

# Disjunctive Invariants

```
counter1: MODULE =
  BEGIN
    LOCAL cnt : INTEGER
    LOCAL b   : BOOLEAN
    INITIALIZATION
      cnt = 0;
      b = TRUE
    TRANSITION
      [       b --> cnt' = (-1 * cnt) - 1;
                    b' = NOT b
        [] ELSE --> cnt' = (-1 * cnt) + 1;
                    b' = NOT b
      ] END;

 Thm2a : THEOREM counter2 |- G(b AND cnt >= 0);
```

Circuit behavior:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| b = | T | F | T | F | T | F | ... |
| cnt = | 0 | -1 | 2 | -3 | 4 | -5 | ... |

Thm2a is our initial approximation ...

# Disjunctive Invariants

. . . And fails

SAL's output:

```
  Counterexample:

Step 0:
--- System Variables (assignments) ---
cnt = 0
b = true
------------------------

Step 1:
--- System Variables (assignments) ---
cnt = -1
b = false
------------------------
```

```
   Thm2b : THEOREM counter2 |- G(   (b AND cnt >= 0)
                                 OR (NOT b AND cnt < 0));
```

Thm2b succeeds.

# Paper Addendum and Challenge

- We were able to complete fully-parameterized proofs of both BMP and the 8N1 Protocol.
- We leave it as a challenge to the real-time model-checking communities, including TReX, HyTech, and Uppal, to reproduce these results for both protocols.

# Final Thoughts on Real-Time Verification Using SMT

We use what Leslie Lamport calls an *explicit-time* model[2] for real-time verification without a real-time model-checker.
Some benefits:

- No new languages and simple semantics.
- SMT is extensible (the theory of arrays, lists, uninterpreted functions, etc.)
- Compositional with non real-time specifications.

[2] *CHARME, 2005*

# Getting our Specifications and SAL

## BMP and 8N1 Specs & Proofs

`http://www.cs.indiana.edu/~lepike/pub_pages/bmp.html`

Google: Brown Pike BMP 8N1

## SRI's SAL

`http://sal.csl.sri.com`

Google: SRI SAL

Also see our *Designing Correct Circuits* paper for improvements.

Thanks to John Rushby, Leonardo de Moura, and our TACAS referees for their comments.

# Timeout Automata[3] (Semantics)

An *explicit* real-time model.

Construct a transition system $\langle S, S^0, \rightarrow \rangle$:

- A set of states $S$, mapping state variables to values.
- A set of initial states $S^0 \subseteq S$.
- A partition on the state variables for $S$, and associated with each partition is a timeout $t \in \mathbb{R}$.
- A set of transition relations, such that $\rightarrow_t$ associated with timeout $t$ and is enabled if for all timeouts $t'$, $t \leq t'$ ($\rightarrow$ is the union of $\rightarrow_t$ for all $t$.)

[3] B. Dutertre and M. Sorea. Timed systems in SAL. *SRI TR*, 2004.

# Parameterized Timing Constraints

SMT allows for *parameterized* proofs of correctness. The following are the parameters from the BMP verification:

```
TIME : TYPE = REAL;

TPERIOD      :     { x : REAL | 0 < x };
TSETTLE      :     { x : REAL | 0 <= x AND x < TPERIOD };
TSTABLE      :     TIME = TPERIOD - TSETTLE;

RSCANMIN :  { x: TIME | 0 < x };
RSCANMAX :  { x: TIME | RSCANMIN <= x AND x < TPERIOD - TSETTLE};
RSAMPMIN :  { x : TIME | TPERIOD + TSETTLE < x };
RSAMPMAX :  { x : TIME | RSAMPMIN <= x AND
                         x < 2 * TPERIOD - TSETTLE - RSCANMAX };
```