IJIT: An API for Boolean Program Analysis with Just-in-Time Translation[1]

Peizun Liu and Thomas Wahl

Northeastern University, Boston, USA

SEFM 2017, Trento, Italy

September 06, 2017

[1] This work is supported by NSF grant no. 1253331.

Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

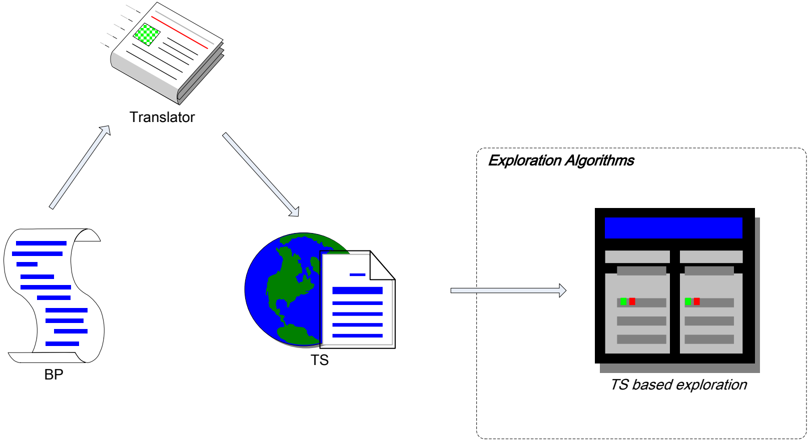
Problem Statement

Boolean program analysis ...

```
decl g1,g2 := *,*;  // global variable
void main() begin
    decl l := 0;    // local  variable
  0: g1,g2 := 0,0;
  1: start_thread 3;
  2: skip;
  3: goto 4, 7;
  4: assume(g1);
  5: l := g1;
  6: goto 8;
  7: assume(!g1);
  8: g1,g2 := !g1,1;
➡ 9: assert(!g2|!l);
end
```

Problem Statement

… through exploration algorithms



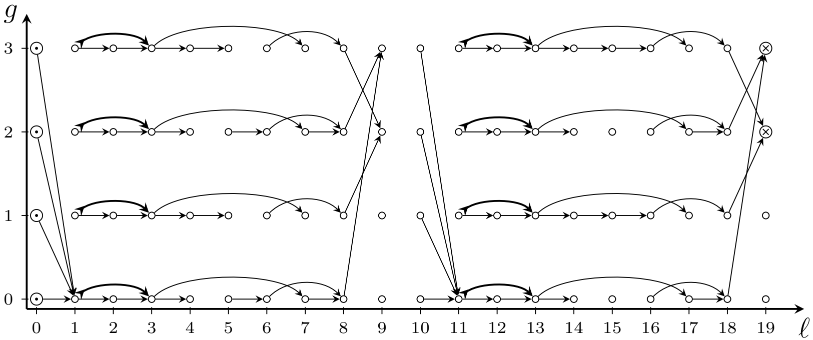
State Space Explosion

For a BP:

```
decl g1,g2 := *,*;  // global variable
void main() begin
    decl l := 0;    // local  variable
  0: g1,g2 := 0,0;
  1: start_thread 3;
  2: skip;
  3: goto 4, 7;
  4: assume(g1);
  5: l := g1;
  6: goto 8;
  7: assume(!g1);
  8: g1,g2 := !g1,1;
➠ 9: assert(!g2|!l);
end
```

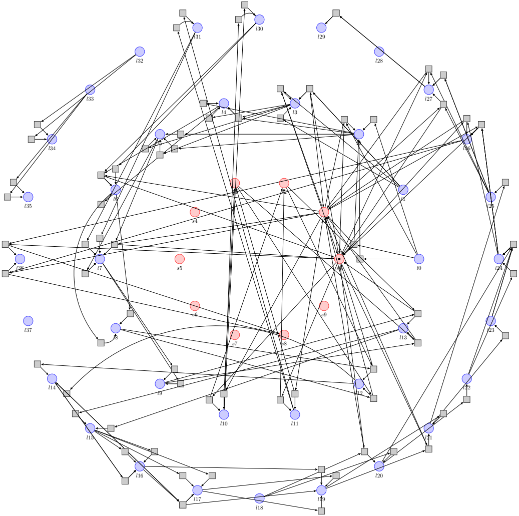
State Space Explosion

From BP to TS (Transition System):



State Space Explosion

From BP to PN (Petri Net):

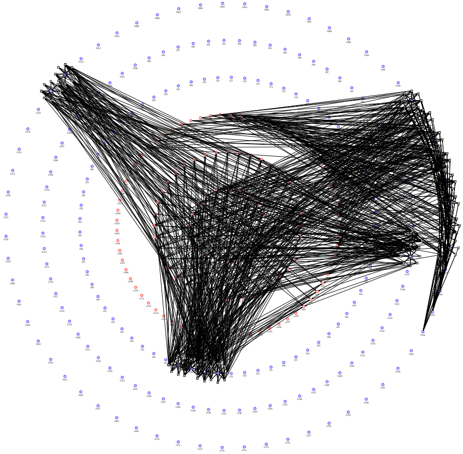


$|R| = 84$

State Space Explosion

From BP to PN: one benchmark

BP: $|V_G| = 5$, $|V_L| = 2$, $LOC = 60$



$|R| = 8064$

Classical Solution: The On-the-Fly Exploration

Advantages:

1. avoids the static transition system construction
2. operates on-the-fly: what you visit is what you pay
3. results in dramatic savings

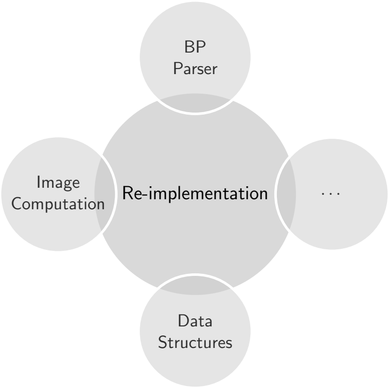
On-the-Fly Exploration: The Problem

... requires re-implementation, which implies

a heavy development burden

On-the-Fly Exploration: The Problem

Is the re-implementation so bad?



An instance:

BWS [Abdulla, 2010]:

	LOC
Original	$\approx$ 1300
Added	$\approx$ 1400
Changed	$\approx$ 600

Updated: $> 150\%$

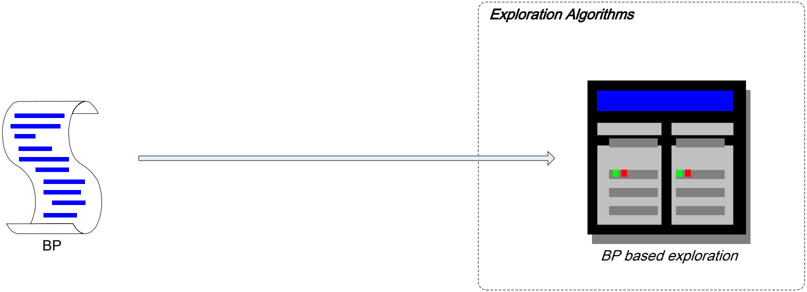
On-the-Fly Exploration: The Problem

... requires re-implementation, which implies

a maintenance nightmare

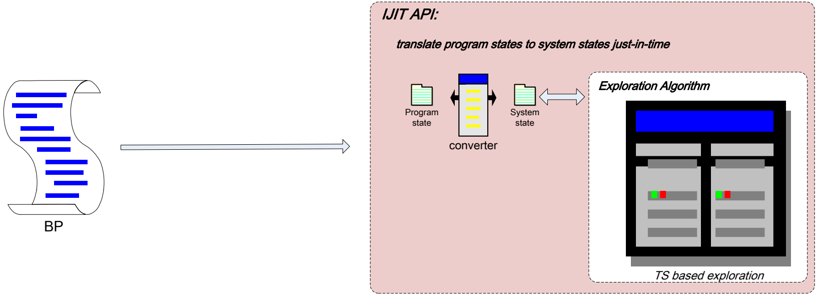
Our Goal

... is to **automate** the “re-implementation”



Our Solution

… is an API for just-in-time translation



Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

BP Analysis with JIT Translation

Target: exploration algorithms

TS Exploration

Scheme 1 EXPLORE(M, T)

Input: transition system M, target T

```
Initialize $W$ and $X$
while $\exists w \in W$
    $W := W \setminus \{w\}$
    for each $w' \in \text{image}(w)$
        if $w'$ not in $X$ then
            if $w'$ in $T$ then
                return "found"
            merge $w'$ into $W$ and $X$
return "not found"
```

BP Analysis with JIT Translation

Approach: an API, with $\mathcal{B} \leftrightarrow M$ conversion functions (f, f^{-1})

JIT Exploration

Scheme 2 $\textsc{Explore\_Ijit}(\mathcal{B}, T)$

Input: Boolean Program $\mathcal{B}$, target T

```
Initialize $W$ and $X$
while $\exists w =\in W$
    $W := W \setminus \{w\}$
    for each $w' \in f^{-1}(\text{image}_{\mathcal{B}}(f(w)))$
        if $w'$ not in $X$ then
            if $w'$ in $T$ then
                return "found"
            merge $w'$ into $W$ and $X$
return "not found"
```

BP Analysis with JIT Translation

Comparison

Before

Scheme 1 EXPLORE(M, T)

```
Input: transition system $M$, target $T$
Initialize $W$ and $X$
while $\exists w \in W$
    $W := W \setminus \{w\}$
    for each $w' \in \text{image}(w)$
        if $w'$ not in $X$ then
            if $w'$ in $T$ then
                return "found"
            merge $w'$ into $W$ and $X$
return "not found"
```

After

Scheme 2 EXPLORE_IJIT($\mathcal{B}$, T)

```
Input: Boolean Program $\mathcal{B}$, target $T$
Initialize $W$ and $X$
while $\exists w =\in W$
    $W := W \setminus \{w\}$
    for each $w' \in f^{-1}(\text{image}_{\mathcal{B}}(f(w)))$
        if $w'$ not in $X$ then
            if $w'$ in $T$ then
                return "found"
            merge $w'$ into $W$ and $X$
return "not found"
```

Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

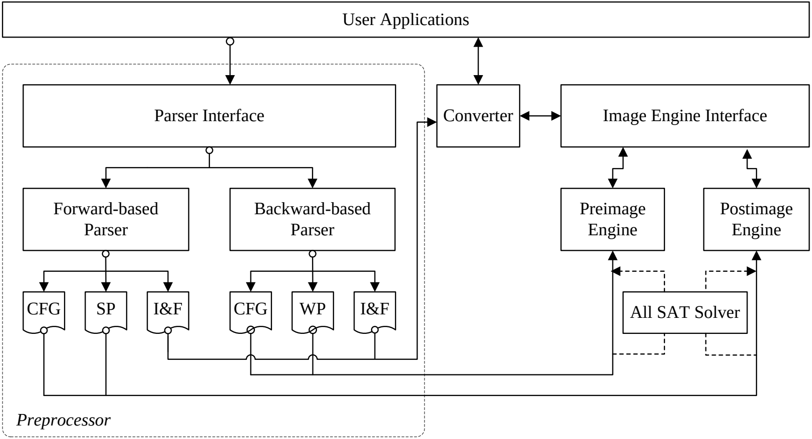
The IJIT API: Usage

Empirical Evaluation

Conclusion

Design

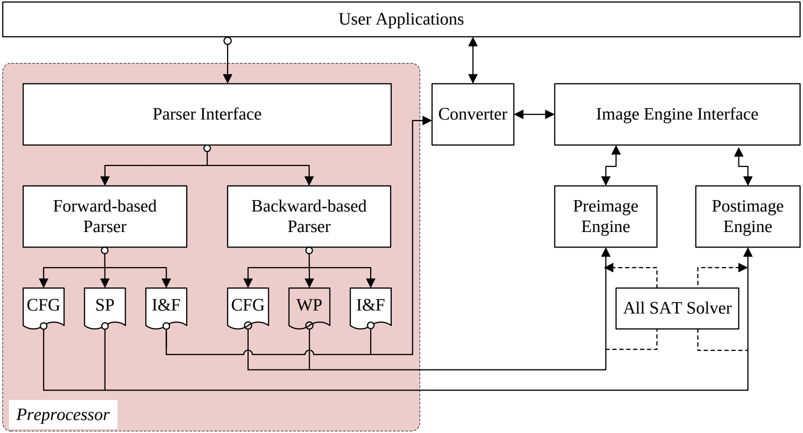
A Schematic Overview of IJIT[2]



[2] IJIT: **I**nterface for **J**ust-**I**n-**T**ime translation.

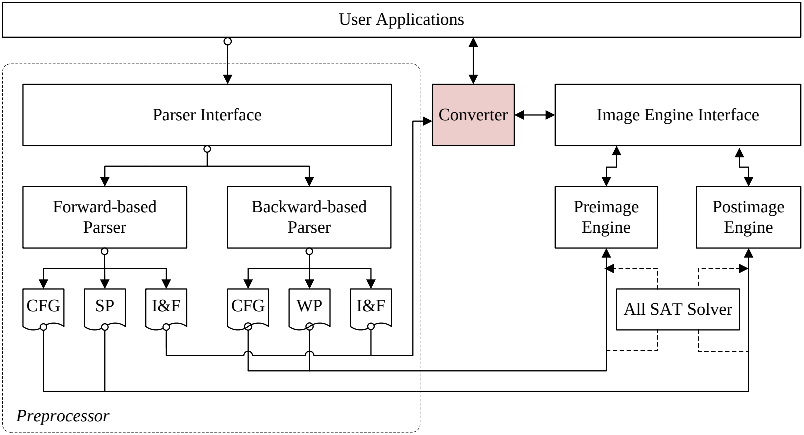
Design

Preprocessor



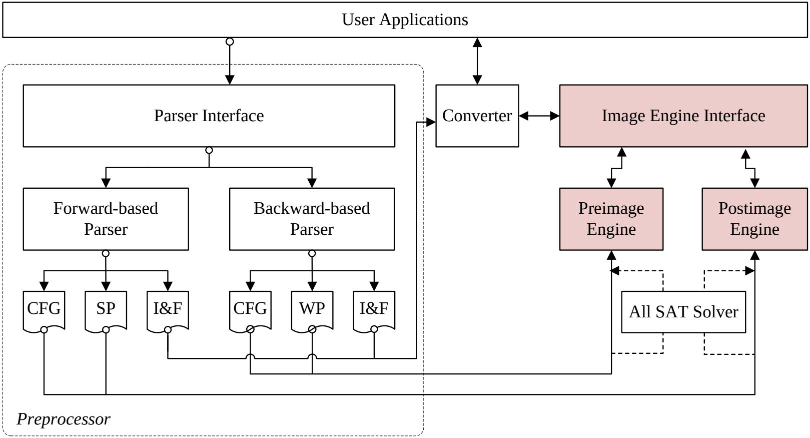
Design

Converter



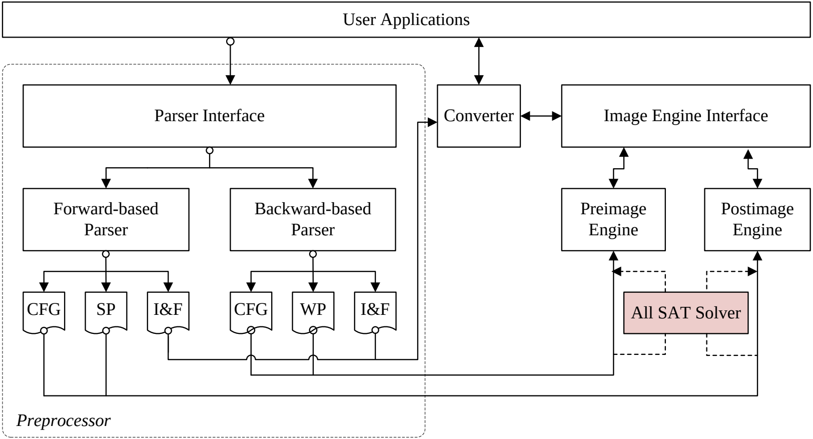
Design

Image Computation



Design

All-SAT solver



Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

API Usage

An example

```
// user's headers, namespace, etc.

void explore() {
  // ...
  auto R = read_file("filename.ts");

  set<state> I = R.init();
  set<state> F = R.final();
  state tau;
  while (...) { // state exploration
    tau = ... ; // an unexplored state
    set<state> Tau = image(tau);

    ...
  } // end while
}
```

API Usage

An example

```
#include "ijit.hh"
using namespace ijit;
void explore_jit() {
  // ...
  auto P = parser::parse("filename.bp", mode::POST);
  converter c;
  set<state> I = c.convert(P.init());
  set<state> F = c.convert(P.final());
  state tau;
  while (...) {
    tau = ... ;
    set<state> Tau = c.convert(
                image(c.convert(tau), mode::POST));
    ...
  }
}
```

API Usage

An example

Before

```
// user's headers, namespace, etc.

void explore() {
  // ...
  auto R = read_file("filename.ts");

  set<state> I = R.init();
  set<state> F = R.final();
  state tau;
  while (...) { // state exploration
    tau = ... ; // an unexplored state
    set<state> Tau = image(tau);

    ...
  } // end while
}
```

After

```
#include "ijit.hh"
using namespace ijit;
void explore_jit() {
  // ...
  auto P = parser::parse("filename.bp", mode::POST);
  converter c;
  set<state> I = c.convert(P.init());
  set<state> F = c.convert(P.final());
  state tau;
  while (...) {
    tau = ... ;
    set<state> Tau = c.convert(
                     image(c.convert(tau), mode::POST));
    ...
  }
}
```

Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

Goal of Evaluation

Evaluate IJIT through comparisons

Version	Trans. BP to TS	How obtained
TS version	statically	original version
JIT version	on the fly	automated with IJIT
BP version	on the fly	manual re-implementation

We expect a performance ranking of the form

$$\textit{BP version} \quad < \quad \textit{JIT version} \quad \ll \quad \textit{TS version}$$

where "$<$" ("$\ll$") means "(much) faster".

Benchmark Algorithms

Backward Search [Abdulla, 2010]

BWS operates over a *well quasi-ordered system* (WQOS).
WQO is the *covers* relation:

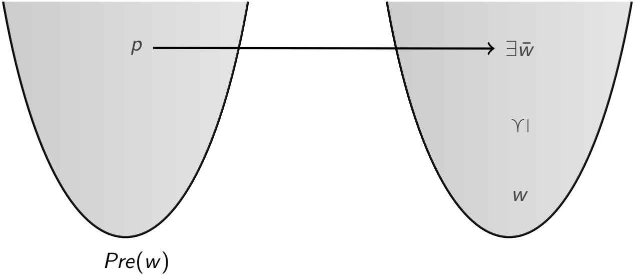
$$(s, \bar{\ell}_1, \ldots, \bar{\ell}_{\bar{n}}) \succeq (s, \ell_1, \ldots, \ell_n)$$

whenever $\quad \mathit{multiset}\{\bar{\ell}_1, \ldots, \bar{\ell}_{\bar{n}}\} \supseteq \mathit{multiset}\{\ell_1, \ldots, \ell_n\}$.

☞ **See paper for more benchmark algorithms and evaluation**

Benchmark Algorithms

Backward Search [Abdulla, 2010]



$$\mathrm{CovPre}(w) \;=\; \min\{p \mid \exists\ \bar{w} :\ p \rightarrow \bar{w} \wedge \bar{w} \succeq w\}$$

Experimental Setup

Benchmark[2]

30 C programs, 155 BPs, we use SatAbs [Clark et al., 2005] to construct the BPs from these programs.

BP	min.	max.
$\|V_G\|$	0	7
$\|V_L\|$	0	7
LOC	29	278

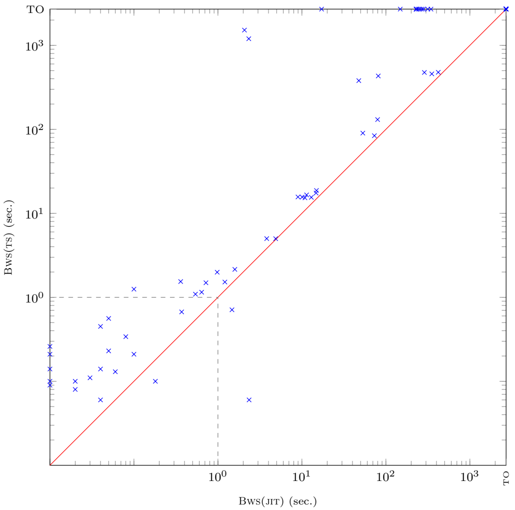
TS	min.	max.
$\|G\|$	5	16385
$\|L\|$	24	3969
$\|R\|$	57	269488

[2] Download me ☺:

- benchmark: *http://www.ccs.neu.edu/home/lpzun/ijit*
- source code: *https://github.com/lpzun/ijit*

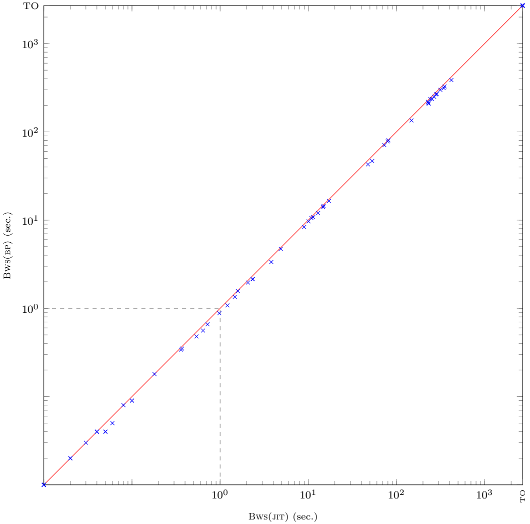
Experimental Evaluation

Evaluation on BWS: Time



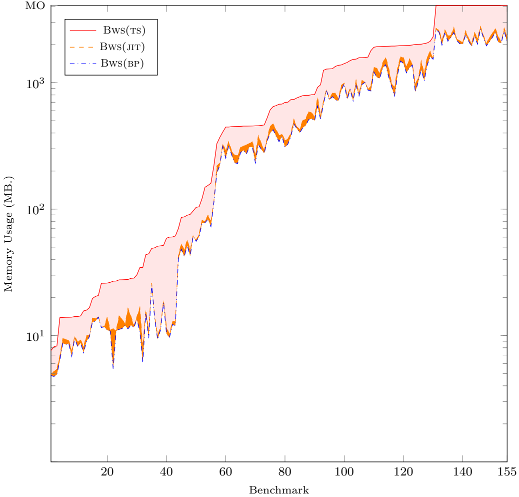
Experimental Evaluation

Evaluation on BWS: Time



Experimental Evaluation

Evaluation on BWS: Memory



Outline

Motivation

BP Analysis with JIT Translation

The IJIT API: Design

The IJIT API: Usage

Empirical Evaluation

Conclusion

Conclusion

Conclusion

- with IJIT, users automatically adapt TS algorithms to BP input;
- with IJIT, users reap almost the same benefits as manual re-implementation.

Future work

- extending IJIT to support Petri Nets;
- extending IJIT to support nonstandard image computations.

Thank You

References

R. M. Karp and R. E. Miller, "Parallel program schemata", *J. Comput. Syst. Sci.*, 1969.

P. A. Abdulla, "Well (and better) quasi-ordered transition systems", *Bulletin of Symbolic Logic*, 2010.

A. Kaiser, D. Kroening, and T. Wahl, "Dynamic cutoff detection in parameterized concurrent programs", *CAV*, 2010.

P. Liu, T. Wahl, "Infinite-state backward exploration of Boolean broadcast programs", *FMCAD*, 2014.

E.M. Clarke, D. Kroening, N. Sharygina, K. Yorav: "SATABS: SAT-based predicate abstraction for ANSI-C", *TACAS*, 2005.