Compositional Proof Rules for Multi-Threaded Program Verification (and their automation)

Corneliu Popeea

- Joint work with Grebenshchikov, Gupta, Lopes
Andrey Rybalchenko -

Technische Universität München

“Verification is solving Horn clauses”

- Our view of the program is a transition system

```
int sum (int i) {
A:  int s = 0;

B:  while (i > 0) {
        s = s + i;
        i = i – 1;
    }

C: assert (s >= 0);
}
```

$V = (pc, s, i)$
$V' = (pc', s', i')$

$Init(V) = (pc = A)$

$Step(V, V') =$
$(pc=A \wedge pc'=B \wedge s'=0 \wedge i'=i) \vee$
$(pc=B \wedge pc'=B \wedge i>0 \wedge s'=s+i \wedge i'=i-1) \vee$
$(pc=B \wedge pc'=C \wedge i\leq 0 \wedge s'=s \wedge i'=i)$

$Error(V) = (pc=C \wedge s<0)$

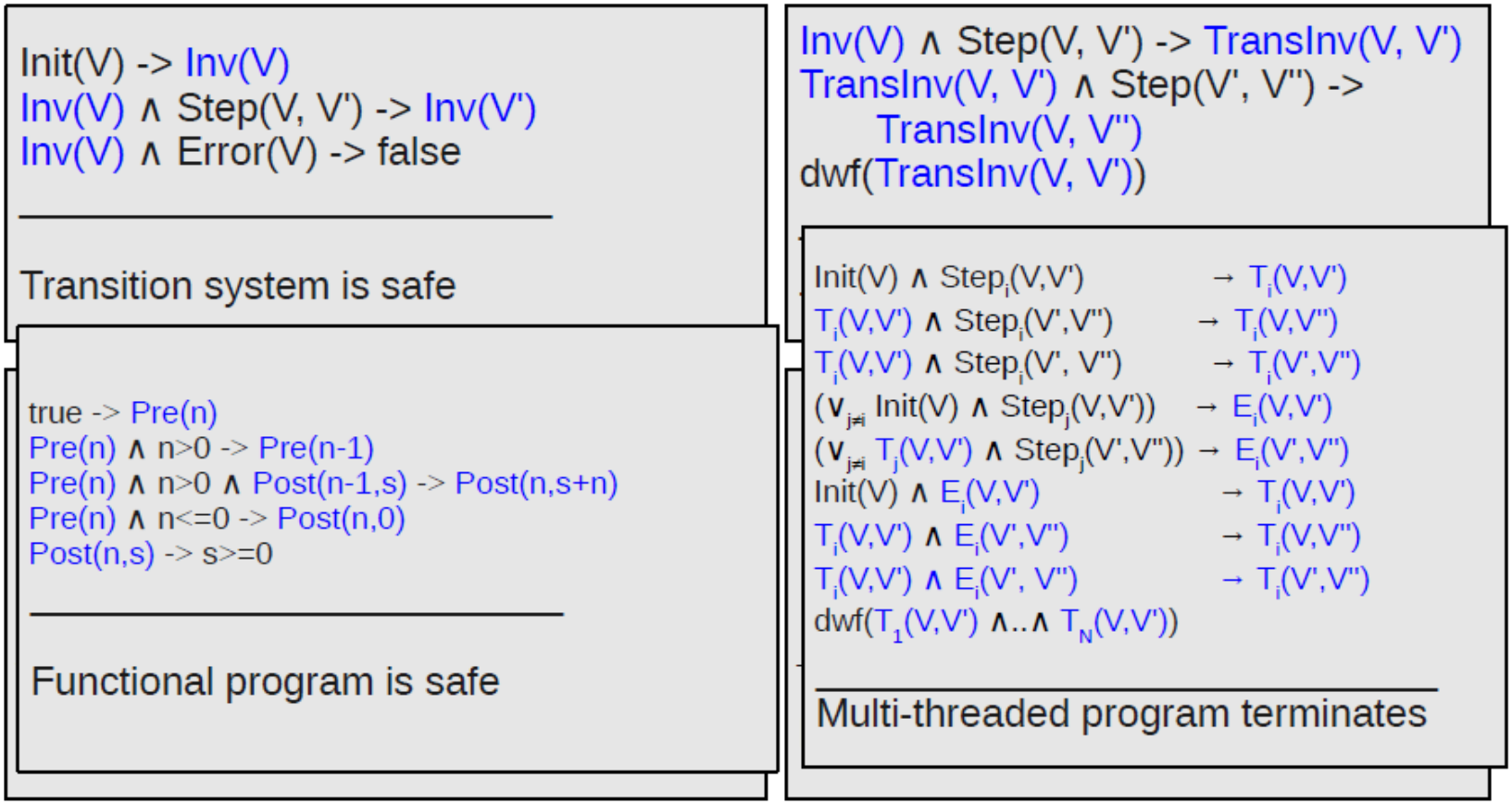
“Verification is solving Horn clauses”

- Find assertion inv(v) such that

$$\text{init}(v) \rightarrow \text{inv}(v)$$
$$\text{inv}(v) \wedge \text{next}(v,v') \rightarrow \text{inv}(v')$$
$$\text{inv}(v) \wedge \text{error}(v) \rightarrow \text{false}$$

Transition system is safe

Safety and termination properties



Safety and termination properties

- Our implementation handles a wide range of verification problems [PLDI'12]
- Verification competitions
  - [TACAS'12] ControlFlowInteger: 91 ok / 93 (2 x time-outs)
  - [TACAS'13] Concurrency: 28 ok / 32 (4 x queue)
- Other tools
  - Blast, CPAchecker, ESBMC, SatAbs, ...

Proof rules for multi-threaded programs

Caveats

- **Sequential consistency**

Pros

- **not bounded context switches**
- **not only thread-modular proofs**
- **not only datarace-free code**

Owicki-Gries proof rule

- Given a transition system,

  N, init(v), next(v,v'), error(v)

- Find assertions $R_1(v)$, .., $R_N(V)$ such that:

  $\text{init}(v) \rightarrow R_i(v)$

  $R_i(v) \wedge \text{next}_i(v,v') \rightarrow R_i(v')$

  $R_i(v) \wedge (\vee_{j \neq i} R_j(v) \wedge \text{next}_j(v,v')) \rightarrow R_i(v')$ } Interference freedom

  $R_i(v) \wedge \text{error}(v) \rightarrow \text{false}$

Concurrent transition system is safe

Rely-guarantee proof of safety

- Given a transition system,

  N, init(v), next(v,v'), error(v)

Environment assumptions

- Find $R_1(v)$, .., $R_N(v)$, $E_1(v,v')$, .. $E_N(v,v')$:

  $init(v) \rightarrow R_i(v)$

  $(\bigvee_j R_j(v) \wedge next_j(v,v')) \rightarrow E_i(v,v')$ } Environment assumptions

  $R_i(v) \wedge (next_i(v,v') \vee E_i(v,v')) \rightarrow R_i(v')$

  $R_1(v) \wedge .. \wedge R_i(v) \rightarrow false$

Concurrent transition system is safe

Rely-guarantee proof of termination

- Given a transition system,
  N, init(v), next(v,v'), error(v)

- Find assertions $T_1(v,v')$, .., $T_N(v,v')$, $E_1(v,v')$, .. , $E_N(v,v')$:

$\text{init}(v) \wedge \text{next}_i(v,v') \rightarrow T_i(v,v')$
$T_i(v,v') \wedge \text{next}_i(v',v'') \rightarrow T_i(v,v'')$
$T_i(v,v') \wedge \text{next}_i(v',v'') \rightarrow T_i(v',v'')$

Reachable computation segments (local)

$(\vee_j \text{init}(v) \wedge \text{next}_j(v,v')) \rightarrow E_i(v,v')$
$(\vee_j T_i(v,v') \wedge \text{next}_j(v',v'')) \rightarrow E_i(v',v'')$

Environment assumptions

$\text{init}(v) \wedge E_i(v,v') \rightarrow T_i(v,v')$
$T_i(v,v') \wedge E_i(v,v') \rightarrow T_i(v',v'')$
$T_i(v,v') \wedge E_i(v',v'') \rightarrow T_i(v,v'')$

Reachable computation segments (environment)

$\text{dwf}(T_1(v,v') \wedge \ldots \wedge T_N(v,v'))$

Well-foundedness check

Concurrent transition system terminates

Example

```
// Thread 1
  lock(l);
  while(x>0) {
    x = x-1;
  }
  unlock(l);

// Thread 2
  while(nondet()) {
    lock(l);
    x = nondet();
    unlock(l);
}
```

- Thread termination for 1?

$$t_1(v,v') = (pc_1 \neq pc_1' \vee l=1 \wedge l'=1 \wedge x'>0 \wedge x'<x)$$

$$E_1(v,v') = (l \neq 1 \vee x'=x)$$

YES

A thread i is terminating, if no computation contains infinitely many steps of i.

Automated verification

Caveats

- **specific theory LI+UIF**

Pros

- **abstraction refinement**
- **bias towards local reasoning**

Solving recursive Horn clauses over LI+UIF

- Finite unfolding of clauses
- Solve recursion-free clauses
- Generalize solution to unbounded unfolding

Tool	CPAchecker 1.1.10-svcomp13				CSeq		ESBMC 1.20		Predator		Threader 0.92	
Limits	timelimit: 900 s, memlimit: 15360 MB											
System	CPU: Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz with 4 cores, frequency: 3401 MHz; RAM: 16343684 kB											
Date of run	2012-11-14 06:53		2012-11-14 08:27		2012-12-04 12:32		2012-11-17 03:16		2012-12-03 03:44		2012-11-12 05:03	
Options	-svcomp13 --skip-recursion --heap 12000m -disable-java-assertions		-svcomp13 --combination --heap 12000m -disable-java-assertions						-t lang1 -m32			
../sv-benchmarks-1.4/	status	time	status	time	status	time	status	time	status	time	status	time
pthread/fib_bench_longer_unsafe.cil.c	unknown	1.2	unknown	1.2	unknown	0.35	unsafe	13	unknown	0.04	unsafe	6.0
pthread/fib_bench_unsafe.cil.c	unknown	1.2	unknown	1.2	unknown	0.07	unsafe	3.5	unknown	0.02	unsafe	4.1
pthread/queue_unsafe.cil.c	unknown	1.4	unknown	1.4	unknown	0.07	timeout	900	unknown	0.08	unknown	86
pthread/reorder_5_unsafe.cil.c	unknown	1.4	unknown	1.4	unknown	0.07	unsafe	120	error	0.10	unsafe	2.6
pthread/twostage_3_unsafe.cil.c	unknown	1.5	unknown	1.4	unknown	0.07	timeout	910	error	0.09	unsafe	3.6
pthread/fib_bench_longer_unsafe.i	unknown	1.4	unknown	1.4	timeout	900	unsafe	9.9	error	0.02	unsafe	5.9
pthread/fib_bench_unsafe.i	unknown	1.4	unknown	1.4	timeout	900	unsafe	2.7	error	0.02	unsafe	4.2
pthread/lazy01_unsafe.i	unknown	1.4	unknown	1.4	unsafe	0.57	unsafe	54	unknown	0.03	unsafe	0.95
pthread/queue_unsafe.i	unknown	1.7	unknown	1.7	unknown	0.10	timeout	910	error	0.03	unknown	0.41
pthread/reorder_2_unsafe.i	unknown	1.7	unknown	1.8	unsafe	16	timeout	910	error	0.04	unsafe	2.2
pthread/reorder_5_unsafe.i	unknown	1.6	unknown	1.6	unknown	0.16	timeout	910	error	0.03	unsafe	2.5
pthread/stack_unsafe.i	unknown	1.5	unknown	1.5	timeout	900	timeout	900	unknown	0.02	unsafe	80
pthread/stateful01_unsafe.i	unknown	1.4	unknown	1.4	unsafe	0.68	unsafe	160	unknown	0.02	unsafe	0.91
pthread/twostage_3_unsafe.i	unknown	1.6	unknown	1.7	unsafe	67	timeout	910	error	0.03	unsafe	14
pthread/fib_bench_longer_safe.cil.c	unknown	1.2	unknown	1.2	unknown	0.07	safe	70	unknown	0.03	safe	6.4
pthread/fib_bench_safe.cil.c	unknown	1.2	unknown	1.2	unknown	0.07	safe	18	unknown	0.02	safe	4.6
pthread/queue_ok_safe.cil.c	unknown	1.4	unknown	1.4	unknown	0.07	unsafe	6.7	unknown	0.10	unknown	58
pthread/fib_bench_longer_safe.i	unknown	1.4	unknown	1.3	timeout	900	safe	52	error	0.03	safe	6.4
pthread/fib_bench_safe.i	unknown	1.4	unknown	1.3	timeout	900	safe	13	error	0.02	safe	4.6
pthread/indexer_safe.i	unknown	1.5	unknown	1.8	safe	0.87	safe	140	error	0.05	safe	4.4
pthread/queue_ok_safe.i	unknown	1.7	unknown	1.7	unknown	0.09	timeout	900	error	0.03	unknown	0.28
pthread/stack_safe.i	unknown	1.5	unknown	1.5	timeout	900	timeout	900	unknown	0.04	safe	250
pthread/stateful01_safe.i	unknown	1.4	unknown	1.4	safe	0.78	safe	540	error	0.02	safe	2.7
pthread/sync01_safe.i	unknown	1.5	unknown	1.5	unknown	0.10	safe	57	unknown	0.03	safe	2.0
pthread-atomic/read_write_lock_unsafe.i	unknown	1.4	unknown	1.3	unsafe	1.8	timeout	920	unknown	0.04	unsafe	2.1
pthread-atomic/dekker_safe.i	unknown	1.4	unknown	1.4	timeout	900	timeout	910	unknown	0.02	safe	3.5
pthread-atomic/lamport_safe.i	unknown	1.4	unknown	1.3	timeout	900	timeout	910	unknown	0.02	safe	35
pthread-atomic/peterson_safe.i	unknown	1.4	unknown	1.4	safe	32	unsafe	27	unknown	0.02	safe	4.9
pthread-atomic/read_write_lock_safe.i	unknown	1.4	unknown	1.4	safe	5.4	timeout	920	unknown	0.02	safe	1.6
pthread-atomic/scull_safe.i	unknown	1.5	unknown	1.5	unknown	0.11	timeout	910	unknown	0.03	safe	99
pthread-atomic/szymanski_safe.i	unknown	1.4	unknown	1.4	safe	140	timeout	910	unknown	0.02	safe	13
pthread-atomic/time_var_mutex_safe.i	unknown	1.4	unknown	1.4	safe	1.2	safe	110	unknown	0.02	safe	4.8
total files	32	45	32	46	32	7500	32	15000	32	1.1	32	720
correct results	0	0	0	0	11	270	15	1400	0	0	28	570
false negatives	0	0	0	0	0	0	0	0	0	0	0	0
false positives	0	0	0	0	0	0	2	34	0	0	0	0
false properties	0	0	0	0	0	0	0	0	0	0	0	0
score (32 files, max score: 49)	**0**		**0**		**17**		**15**		**0**		**43**	

SV-COMP 2013

- Tools:
  - CSeq
  - ESBMC
  - Threader
- Threader:
- No reds – no incorrect results wrt. to the assumptions
- Only 4 unknowns
- Proofs of correctness and counterexamples
- Fastest

SV-COMP 2012

- Tools:
  - ESBMC
  - SatAbs

Tool	CPAchecker ABE r4569		CPAchecker ABM r4573		ESBMC 1.17		FShell 1.3		Predator		SatAbs 3.0	
Limits	timelimit: 900 s, memlimit: 15000 MB											
System	os: Linux 2.6.35-30-generic x86_64 cpu: Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz cores: 4, frequency: 3401 MHz, ram: 16375440 kB											
Date of run	2011-12-03 10:32		2011-12-04 14:36		2011-12-04 08:46		2011-12-05 01:14		2011-12-04 23:44		2011-12-05 13:41	
Benchmark	concurrency		concurrency		concurrency		concurrency		concurrency		concurrency	
Options	-heap 12500m -sv-comp12		-heap 12500m -sv-comp12-abm		--64 --error-label ERROR --no-bounds-check --no-div-by-zero-check --no-assertions --no-pointer-check --no-unwinding-assertions --partial-loops --unwind 7		--unwind 10 --query-file benchmarks/fshell_query --no-unwinding-assertions --32		-m32		--full-inlining --iterations 500 --error-label ERROR --max-threads 5 --modelchecker boom --concurrency --32	
../sv-benchmarks/pthread/	status	runtime	status	runtime	status	runtime	status	runtime	status	runtime	status	runtime
fib_bench_BUG.cil.c	unknown	1.8	unknown	1.8	unsafe	14	error	0.14	unknown	0.77	timeout	910
fib_bench_longer_BUG.cil.c	unknown	1.4	unknown	1.8	unsafe	68	error	0.06	unknown	0.04	timeout	910
queue_BUG.cil.c	unknown	1.5	unknown	1.7	unsafe	0.18	error	0.07	unknown	0.04	failure	0.22
reorder_5_BUG.cil.c	unknown	1.9	unknown	1.6	unsafe	38	error	0.07	unknown	0.04	unsafe	1.4
twostage_3_BUG.cil.c	unknown	1.9	unknown	1.6	**safe**	900	error	0.08	unknown	0.04	failure	0.11
fib_bench.cil.c	unknown	1.4	unknown	1.4	safe	25	error	0.06	unknown	0.03	timeout	910
fib_bench_longer.cil.c	unknown	1.8	unknown	1.4	safe	120	error	0.07	unknown	0.03	timeout	910
queue_ok.cil.c	unknown	1.5	unknown	1.9	safe	0.15	error	0.07	unknown	0.04	failure	0.16
total files	8	13	8	13	8	1200	8	0.62	8	1.0	8	3600
correct results	0	0	0	0	7	270	0	0	0	0	1	1.4
false negatives	0	0	0	0	1	900	0	0	0	0	0	0
false positives	0	0	0	0	0	0	0	0	0	0	0	0
score (8 files, max score: 11)	**0**		**0**		**6**		**0**		**0**		**1**	

Conclusion

- Compositional reasoning is useful
- Minimal heap reasoning required
  (some mutexes allocated in the heap)
- Universal and existential arrays properties
- Relevancy of benchmarks

Extra Slides

Example: “Fibonacci”

```
#include <pthread.h>

int i=1, j=1;
#define NUM 6
void *t1(void* arg){
  int k = 0;
  for (k = 0; k < NUM; k++)    i+=j;
}

void *t2(void* arg){
  int k = 0;
  for (k = 0; k < NUM; k++)    j+=i;
}
```

```
int main(int argc, char **argv){
  pthread_t id1, id2;
  pthread_create(&id1, NULL, t1, NULL);
  pthread_create(&id2, NULL, t2, NULL);
  assert(i <= 377 && j <= 377);
  return 0;
}
```

$$R_1(v) = (i \leq 1 \wedge i \leq 2 \wedge i \leq 3 \wedge \ldots \wedge i \leq 377)$$

$$R_2(v) = (i \leq 1 \wedge i \leq 2 \wedge i \leq 3 \wedge i \leq 5 \wedge \ldots)$$

```
#define SIZE (5)
#define OVERFLOW  (-1)
#define UNDERFLOW (-2)

static int top=0, arr[SIZE];
pthread_mutex_t m;

void inc_top(void){  top++;}
void dec_top(void){  top--;}
int get_top(void){  return top;}
int stack_empty(void){  (top==0) ? TRUE : FALSE; }

int push(unsigned int *stack, int x){
  if (top==SIZE)   {
    printf("stack overflow\n");    return OVERFLOW;
  } else { stack[get_top()] = x; inc_top();  }
  return 0;
}

int pop(unsigned int *stack){
    if (top==0)   {
        printf("stack underflow\n");
        return UNDERFLOW;
    } else { dec_top();  return stack[get_top()];
    return 0;
}
```

```
void *t1(void *arg) {
    int i;  unsigned int tmp;  for(i=0; i<SIZE; i++)
    pthread_mutex_lock(&m);
    tmp = nondet() % SIZE;
    if ((push(arr,tmp)==OVERFLOW))  assert(0);
    pthread_mutex_unlock(&m);
}}

void *t2(void *arg) {
    for(int i=0; i<SIZE; i++)  {
        pthread_mutex_lock(&m);
        if (top>0)  {
            if (pop(arr)==UNDERFLOW))  assert(0);
        }
        pthread_mutex_unlock(&m);
}}

int main(void) {
        pthread_t id1, id2;
        pthread_mutex_init(&m, 0);
        pthread_create(&id1, NULL, t1, NULL);
        pthread_create(&id2, NULL, t2, NULL);
        pthread_join(id1, NULL);
        pthread_join(id2, NULL);
        return 0;
}
```

stack_safe.c

References

[POPL'11]	Solving rec.-free over LI	with Gupta
[ATVA'10]	Control abstraction	with Gupta
[APLAS'11]	Solving rec.-free over LI+UF	with Gupta
[TACAS'12]	Termination proof rules	
[PLDI'12]	Solving recursive Horn clauses	with Grebenshchikov, Lopes
[CAV'11]	Threader – tool paper	with Gupta
[TACAS'12]	HSF – competition contribution	with Grebenshchikov, Lopes
[TACAS'13]	Threader – competition contribution	

all papers joint work with Rybalchenko