Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

The ComFoRT Reasoning Framework

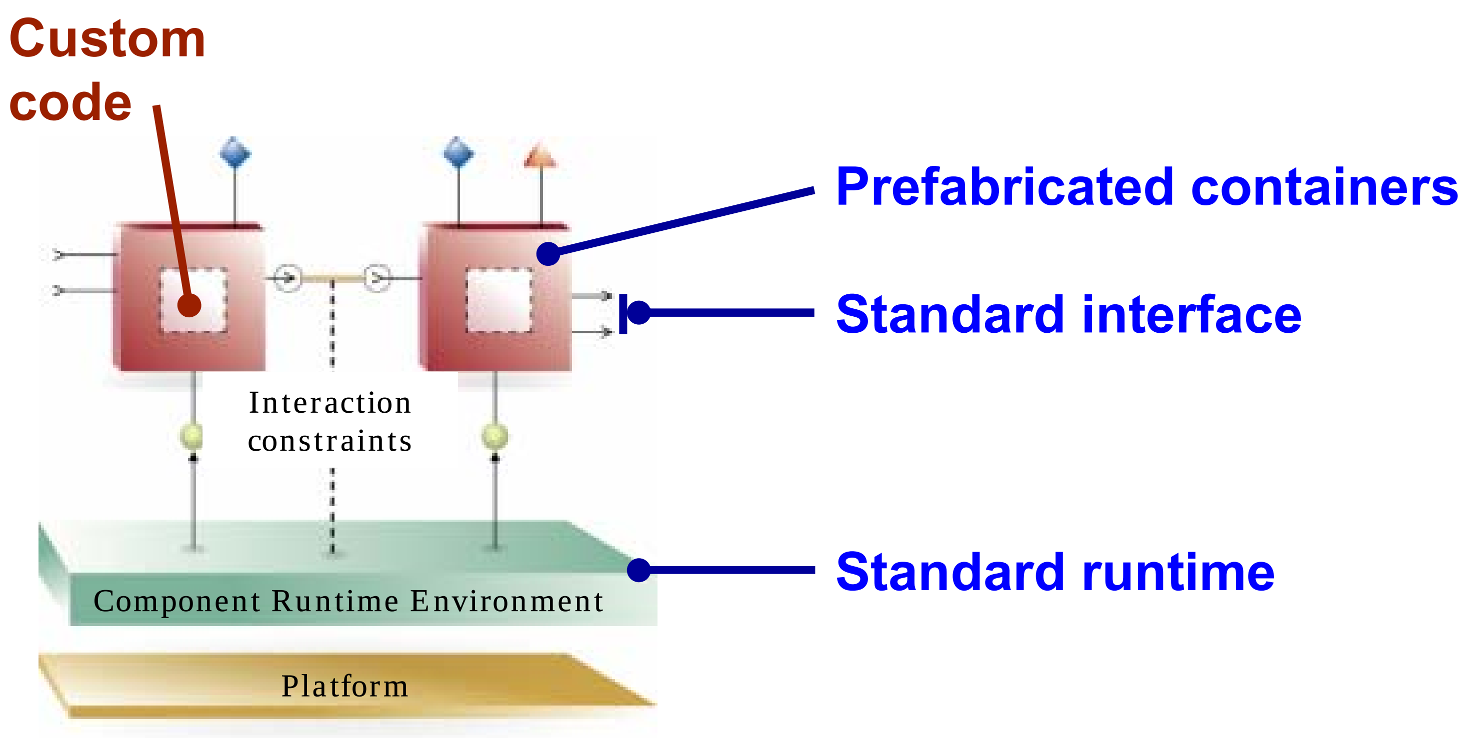
Sagar Chaki
James Ivers
Natasha Sharygina
Kurt Wallnau

Predictable Assembly from Certifiable Components

Enable the development of software systems from software components where:

- critical runtime attributes e.g., performance and safety, are reliably predicted (**predictable assembly**)
- properties of software components needed for prediction are trusted (**certifiable components**)

PACC Component Technology Idiom

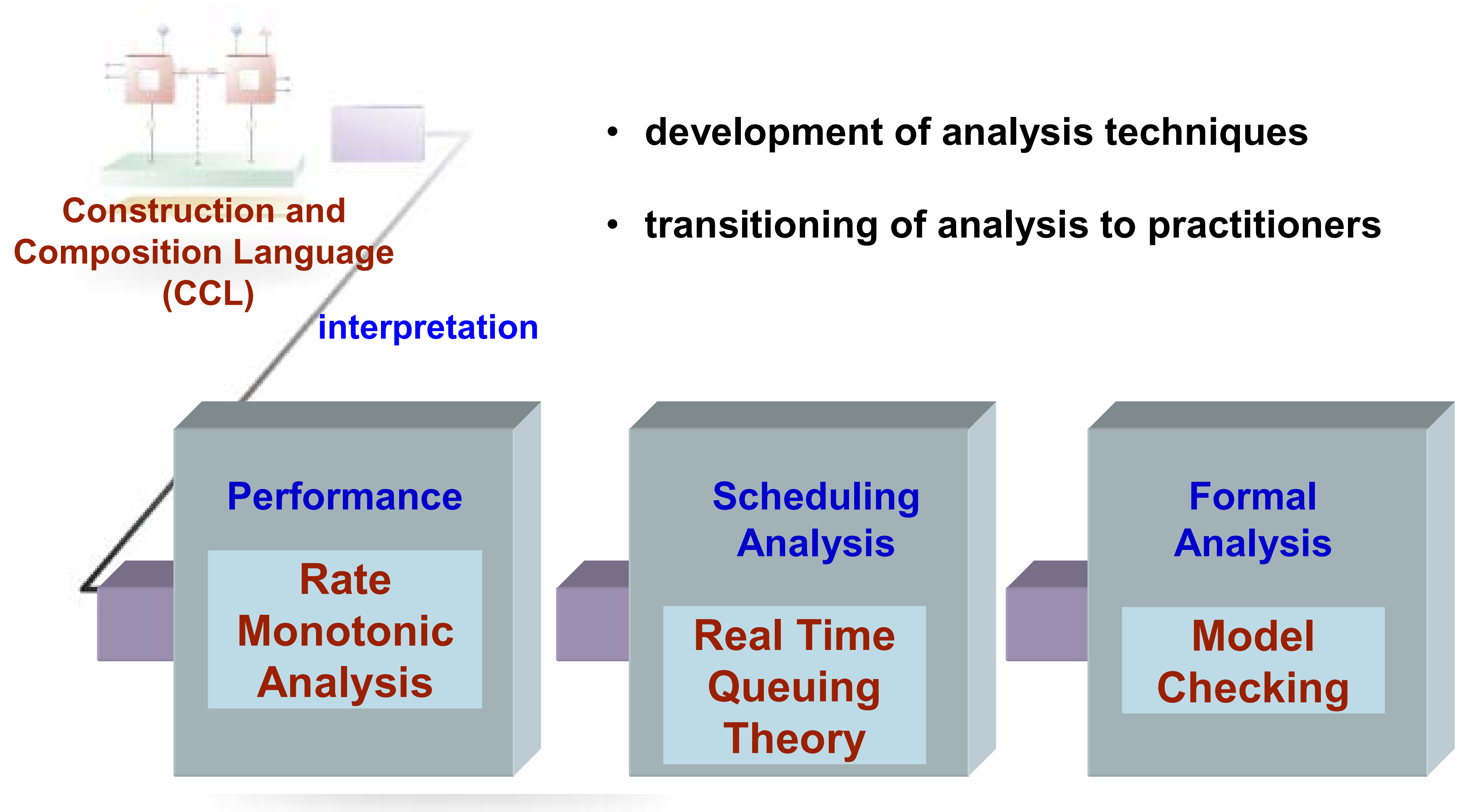


The Construction and Composition Language (CCL) formalizes this idiom

Carnegie Mellon
Software Engineering Institute

PACC Reasoning Frameworks

- development of analysis techniques
- transitioning of analysis to practitioners



ComFoRT Reasoning Framework

- Contains a software model checker Copper:
  - provides new model checking techniques developed for verification of component software
  - builds on academic tool MAGIC
- Analysis models are automatically extracted from programs
- Claims and verification results (counterexamples) are mapped to programs

Verification Domain

High-level designs (CCL programs) and C programs

- Sequential and concurrent

Communication via shared actions

- Synchronous communication
- Asynchronous execution

Copper Capabilities

State/Event-based Verification

- leverages distinction between *data* and *communication actions*

Compositional Deadlock Detection

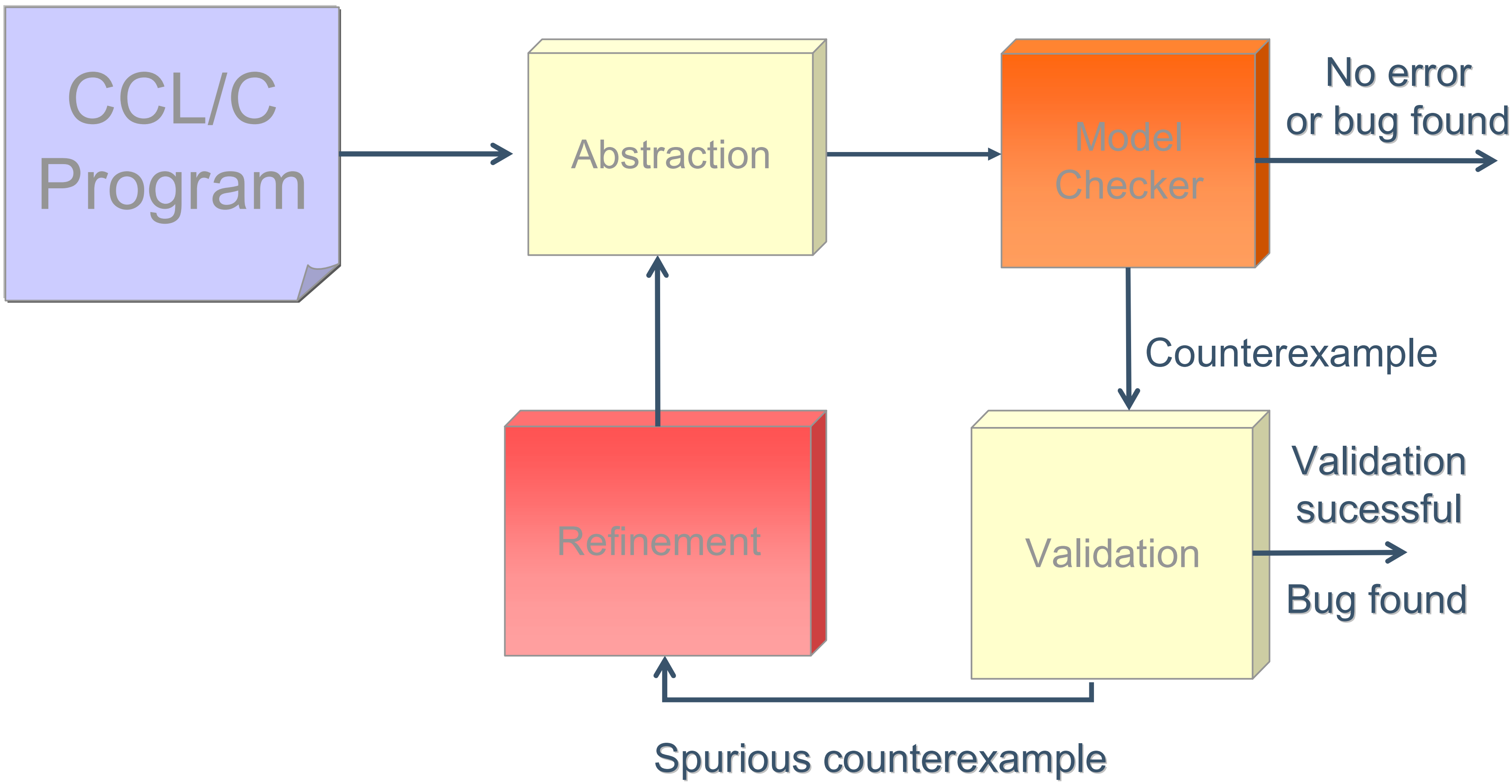
- automated deadlock detection that ensures *sound abstractions* and acts as a space reduction procedure

Verification of Evolving Systems

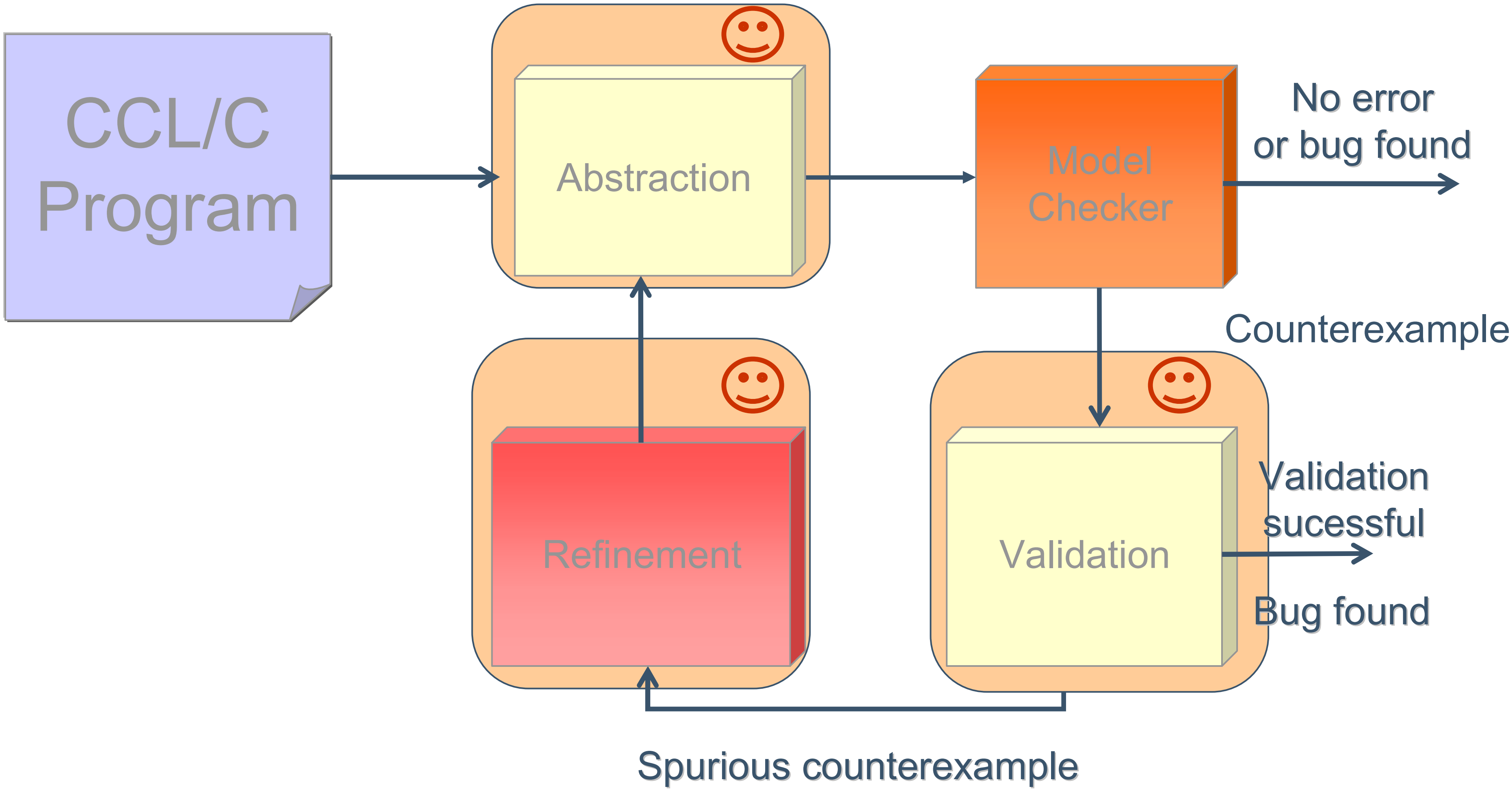
- automated component *substitutability checks*

Carnegie Mellon
Software Engineering Institute

ComFoRT Underlying Framework



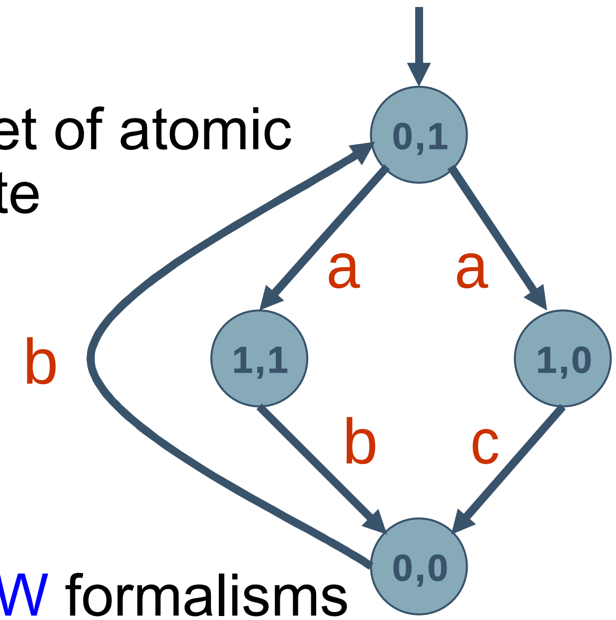
ComFoRT Underlying Framework



State/Event-based Model Checking (IFM04)

Labeled Kripke Structures

- Every state is labeled with a set of atomic propositions, P, true in the state
- Every LKS comes with an alphabet of actions, Σ



State/Event LTL and State/Event AW formalisms

Efficient model checking algorithms for SE-LTL and SE-AW employing the compositional abstraction-refinement framework

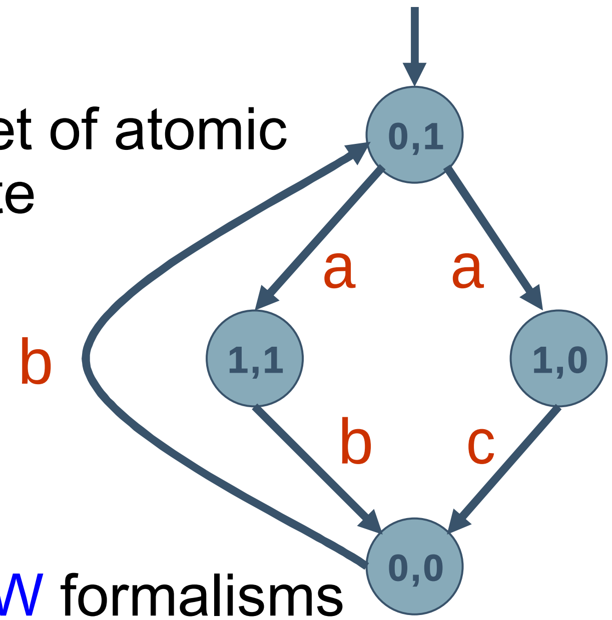
Carnegie Mellon
Software Engineering Institute

State/Event-based Model Checking

Labeled Kripke Structures

- Every state is labeled with a set of atomic propositions, P, true in the state

- Every LKS comes with an alphabet of actions, Σ

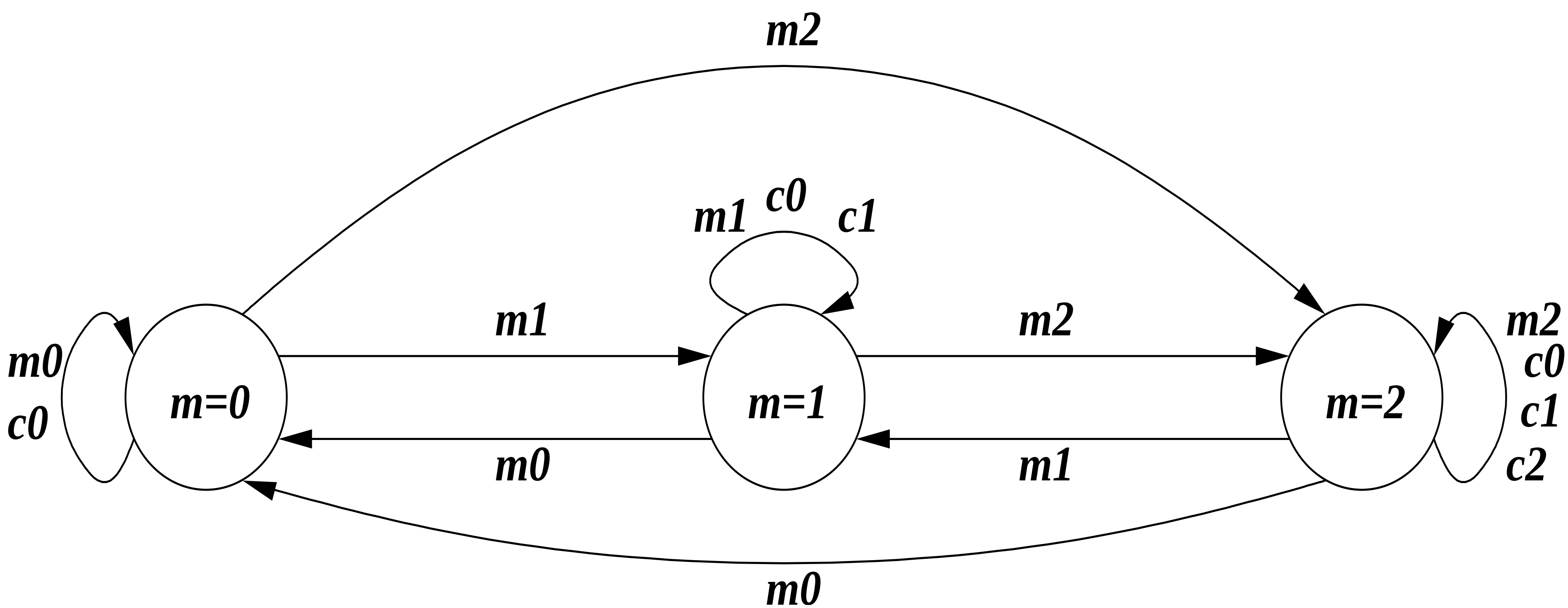


State/Event LTL and State/Event AW formalisms

State/Event-based Software Model Checking, In Proceedings of IFM *Integrated Formal Methods 2004 Conference,* by Sagar Chaki, Edmund Clarke, Joel Ouaknine, Natasha Sharygina and Nishant Sinha.

Carnegie Mellon
Software Engineering Institute

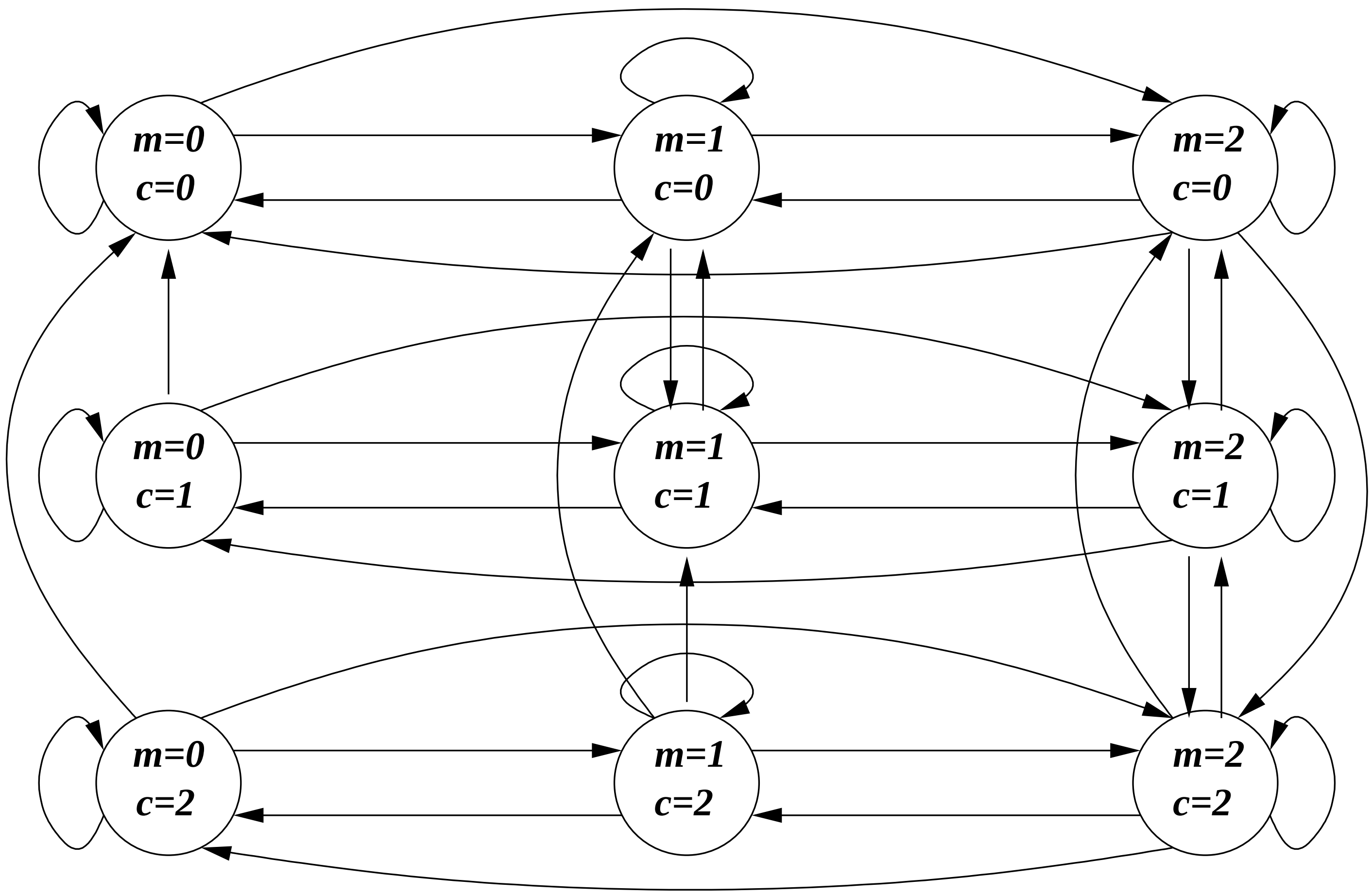
Surge Protector : State/Event



Changes of current beyond threshold are disallowed

G ((c2 → m=2) & (c1 → (m=1 V m=2)))

Surge Protector : State Only



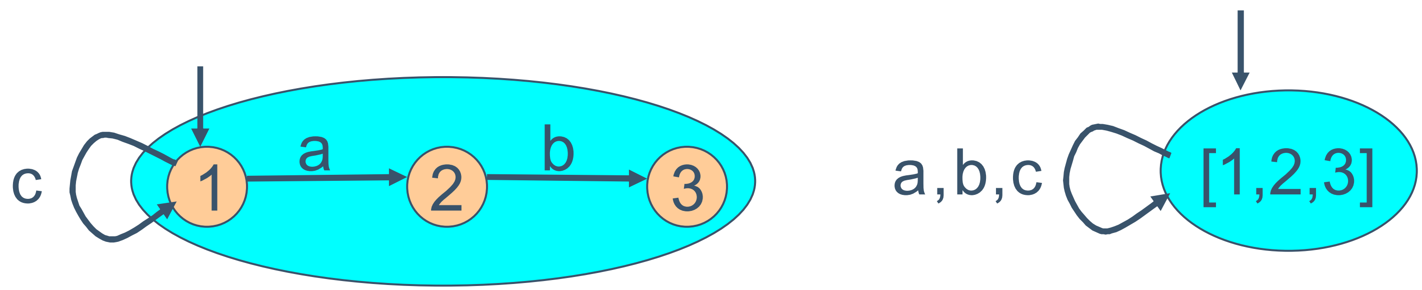
G (((c=0 V c=2) & **X** (c=1)) → (m=1 V m=2)) &

G (((c=0 V c=1) & **X** (c=2)) → m=2)

Deadlock Detection (MEMOCODE'04)

Deadlocks are not preserved by abstraction

- Abstraction refinement does not work



{b} {a,c} {a,b,c} AbsRef={a,b,c}

Copper: Deadlock=AbsRef(s) = Σ

to preserve deadlock the *abstract model* over-approximates not just what *concrete program* can do but also what it refuses

Carnegie Mellon
Software Engineering Institute

Compositional Deadlock Detection

Deadlock is inherently non-compositional

- Can't say anything by looking at components individually

Copper: $AbsRef(A_1, A_2) = AbsRef(A_1) \cup AbsRef(A_2)$

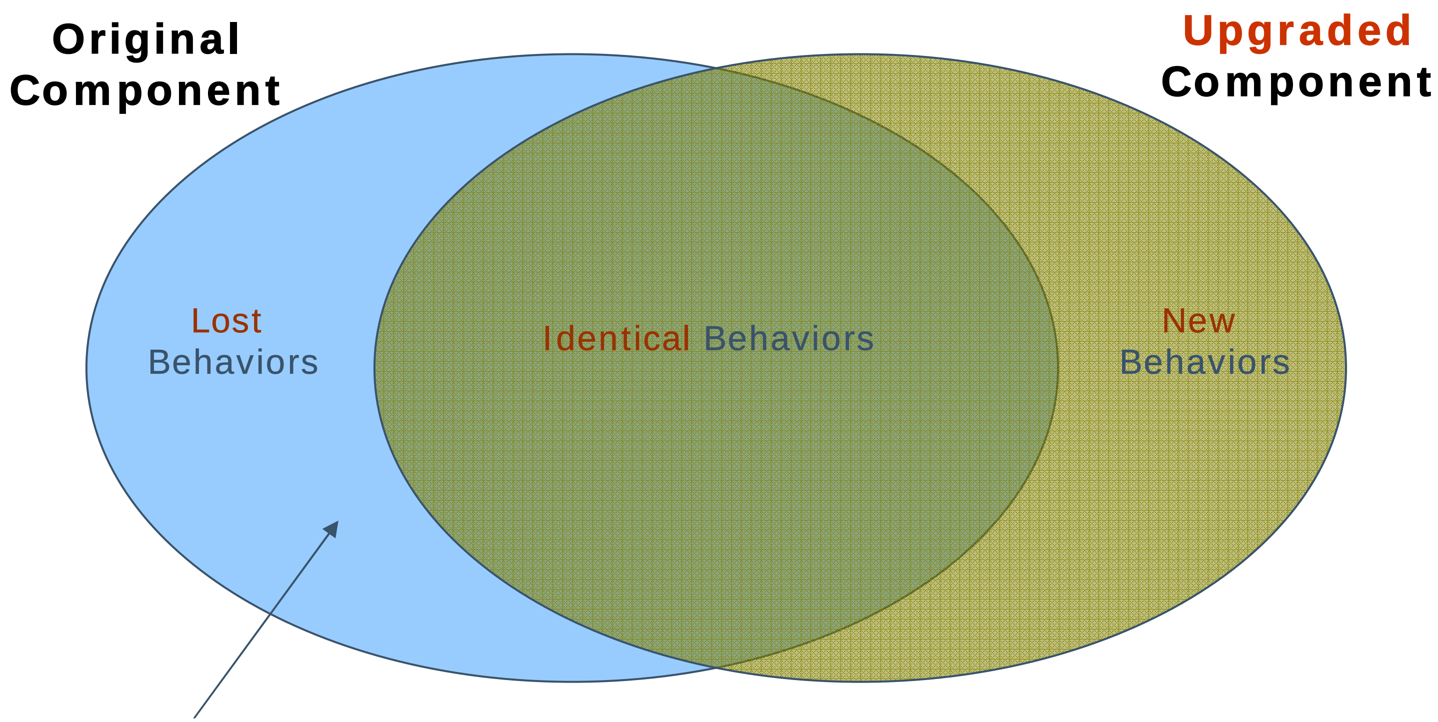
Abstract deadlock - reachable state s such that AbsRef(s) = Σ

Copper: No abstract deadlock in abstract models   No deadlock in concrete models

Automated, compositional and iterative deadlock detection, In Proceedings of the *Conference on Formal Methods for Codesign (MEMOCODE) 2004*, by Sagar Chaki, Edmund Clarke, Joel Ouaknine and Natasha Sharygina

Carnegie Mellon
Software Engineering Institute

Component Substitutability Check

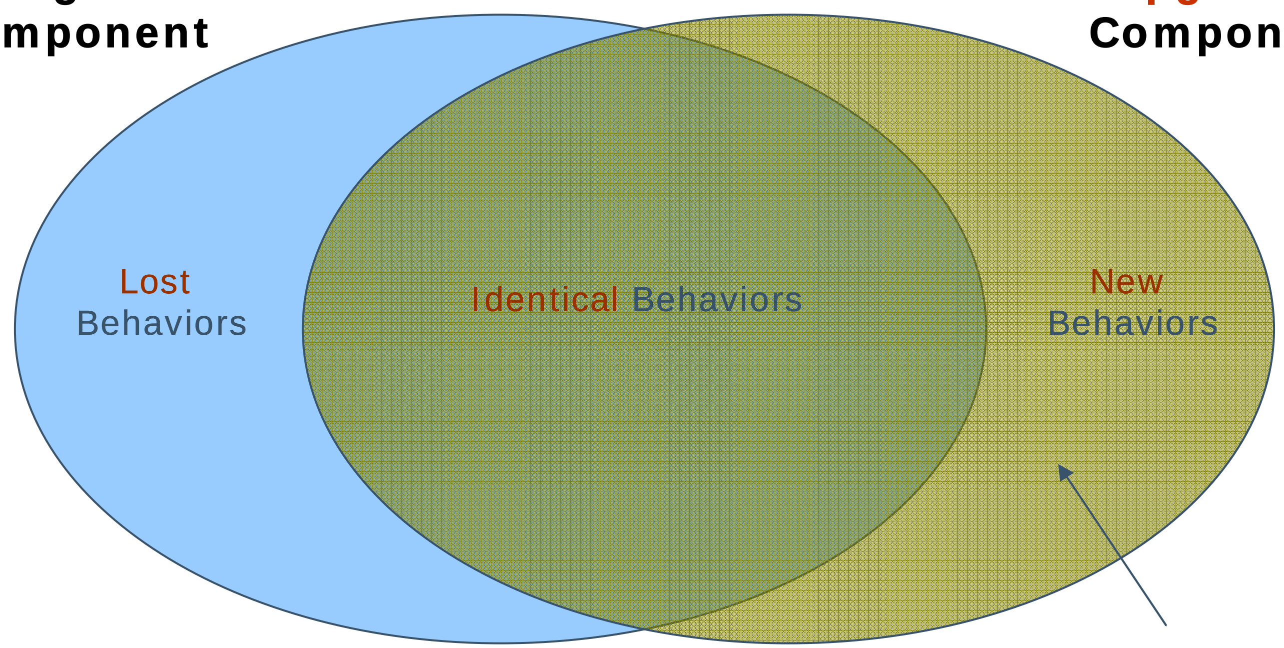


Containment check (Local correctness)

Are all local old services (properties) of the verified component contained in the upgraded component?

Carnegie Mellon
Software Engineering Institute

Component Substitutability Check



Compatibility Check (Global safety check)

Are new services of the upgraded component safe with respect to other components in assembly: all global specifications still hold?

Substitutability Check Approach

- Procedure for checking *simultaneous upgrades* of *multiple components* (FM'04)

  - Abstraction (under- and over- approximations) for the component containment check

  - Compositional reasoning + learning regular sets for automated compatibility check

- Procedure for checking *individual component upgrades* (SAVCBS'04)

  - Algorithms based on learning regular sets technique for the component containment and compatibility tests

Substitutability Check Approach

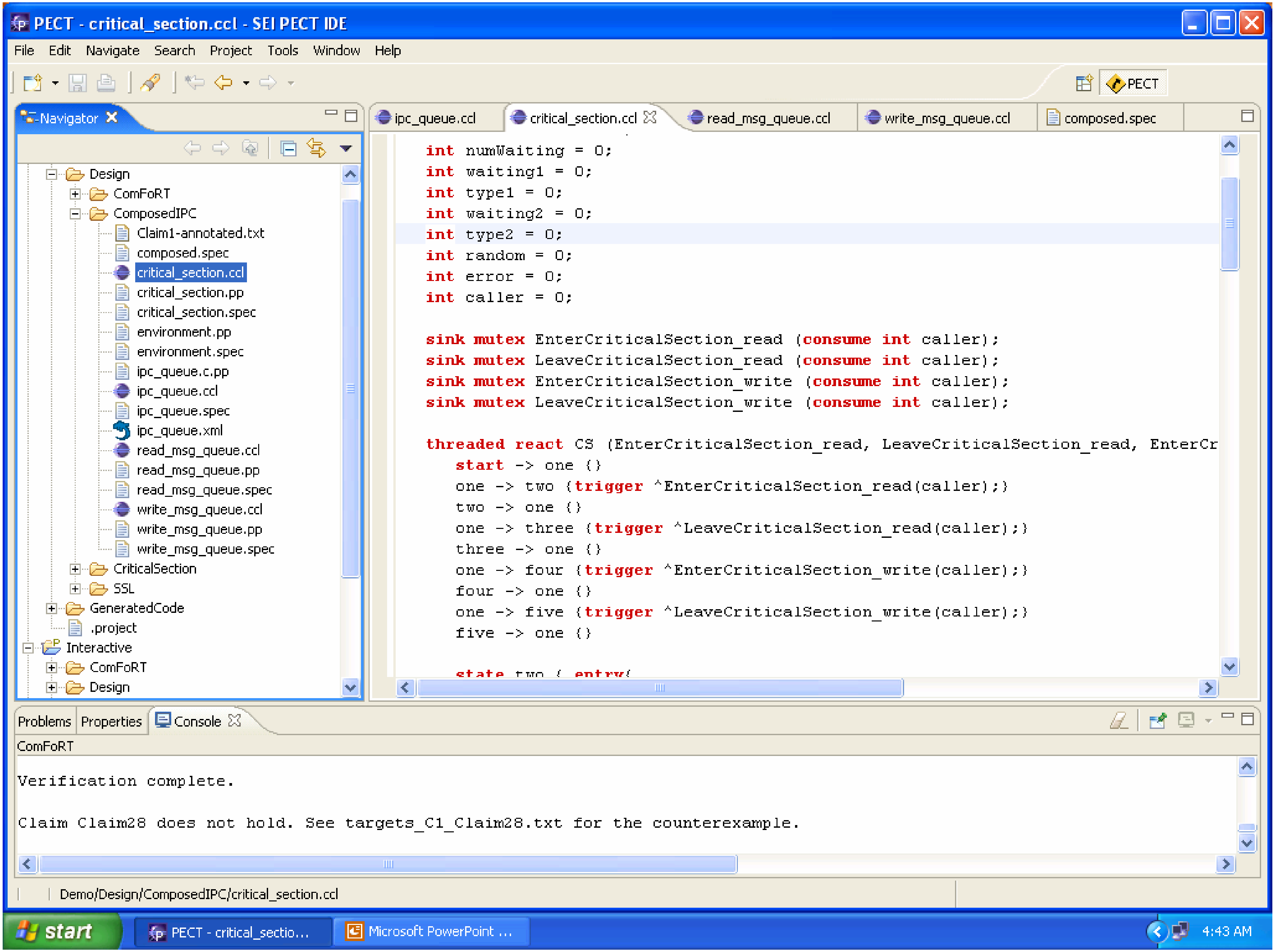
- Procedure for checking *simultaneous upgrades* of *multiple components*

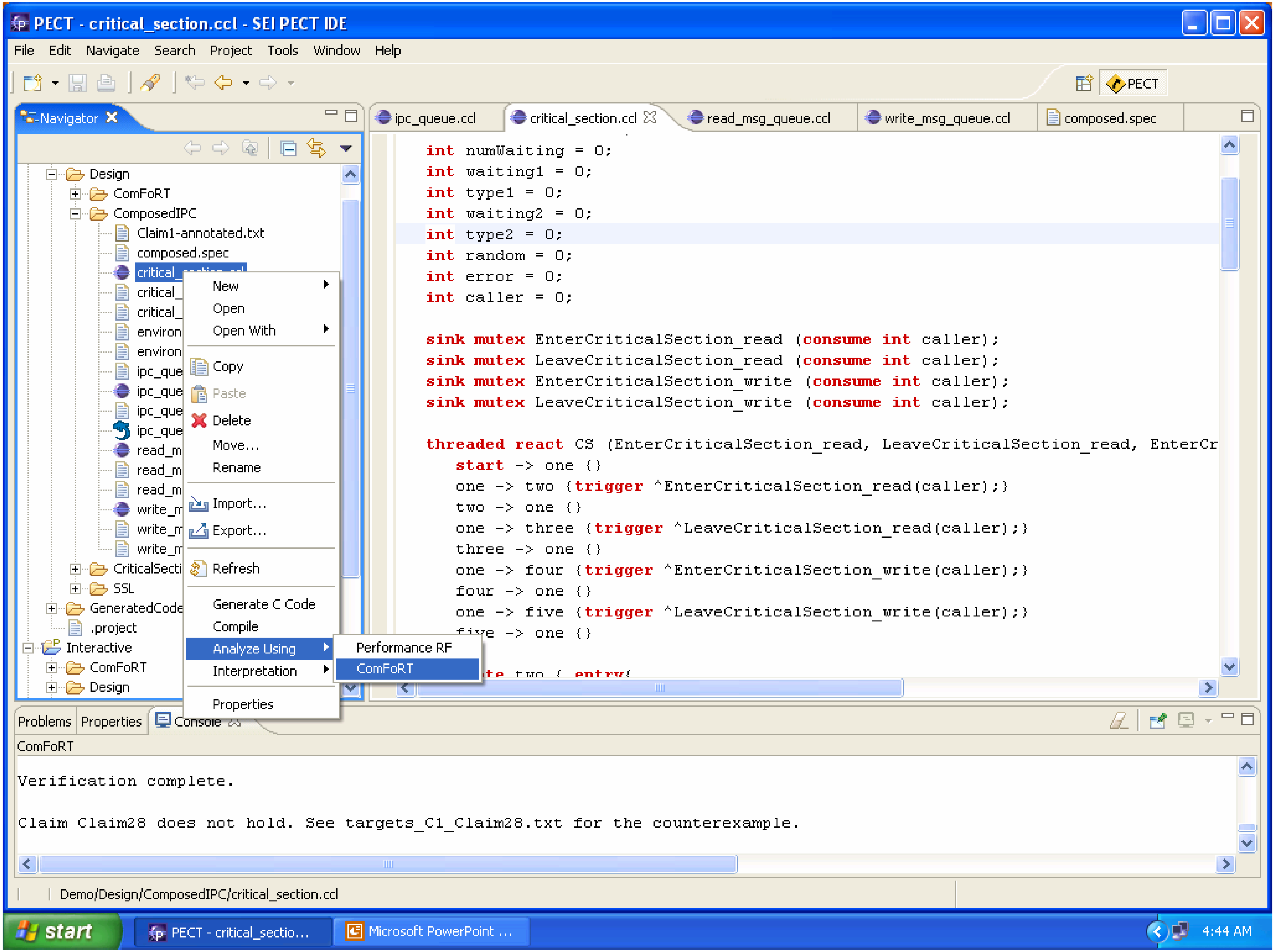
  - Abstraction (under- and over- approximations) for the component containment check

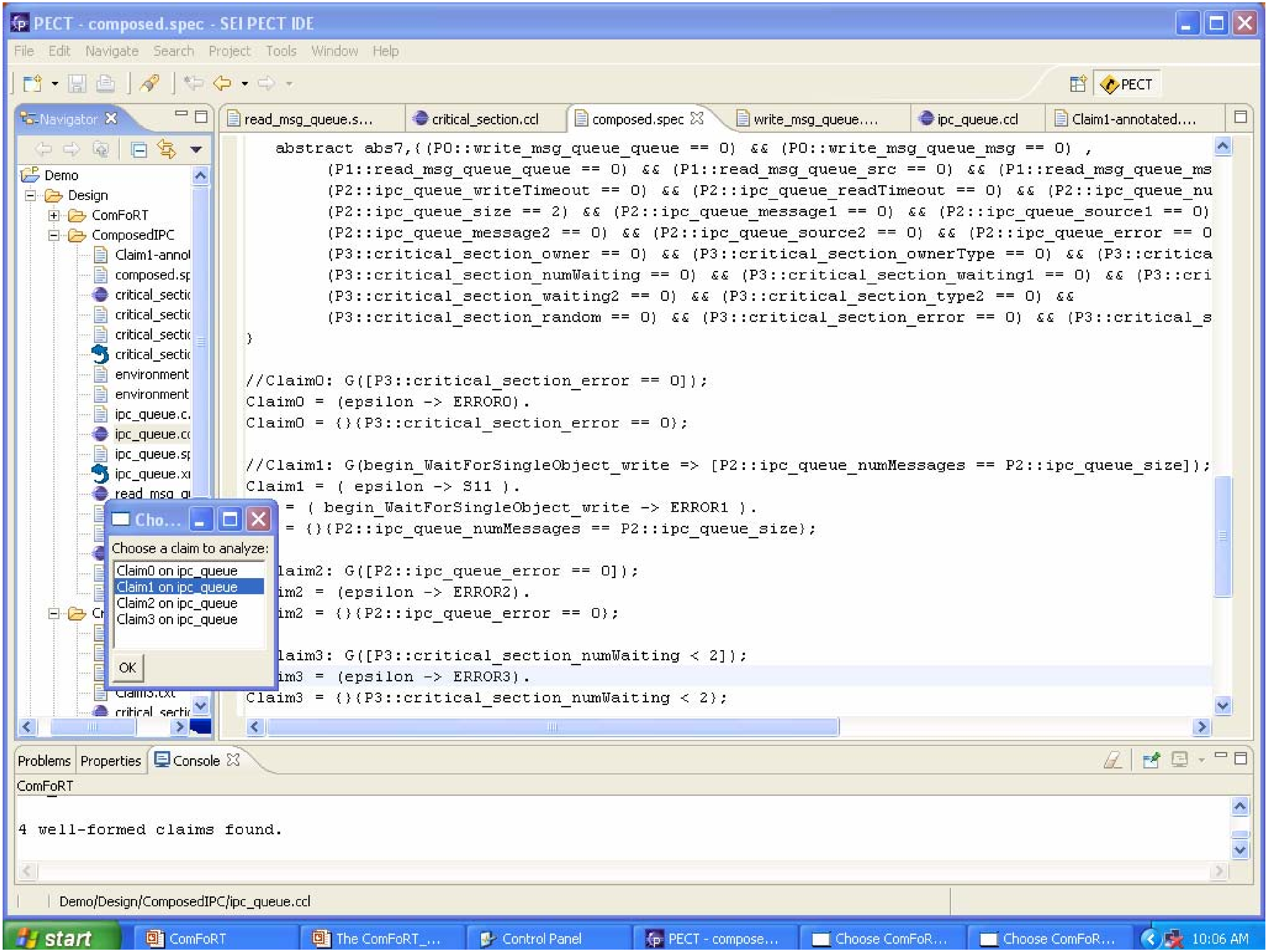
  - Compositional reasoning + learning regular sets for automated compatibility check

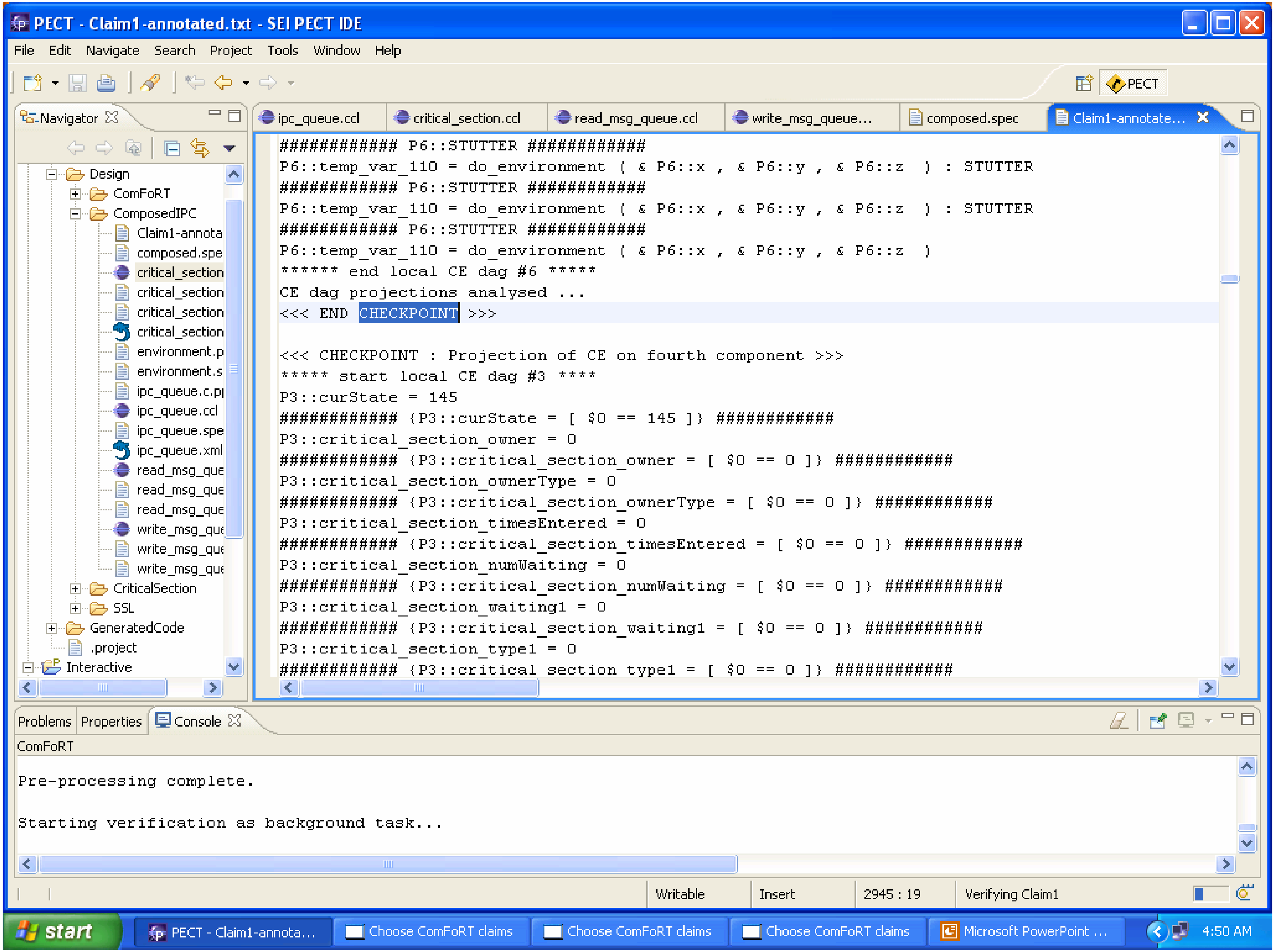
Dynamic Component Substitutability Analysis, In Proceedings of FM 2005 *Formal Methods Conference, by Sagar Chaki,* Ed Clarke, Natasha Sharygina and Nishant Sinha.

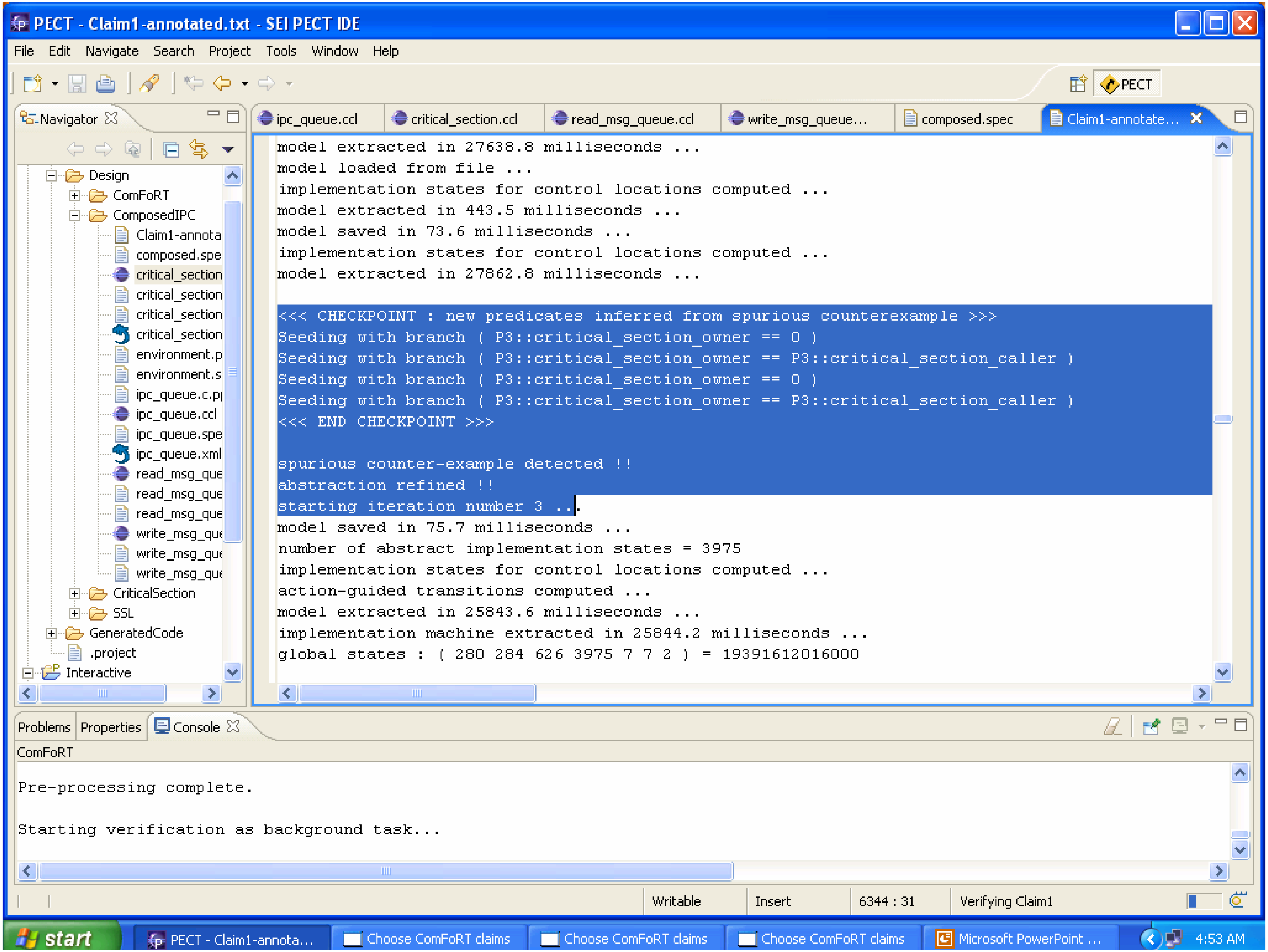
Verification of Evolving Software, In Proceedings of SAVCBS 2004 by Sagar Chaki, Natasha Sharygina and Nishant Sinha

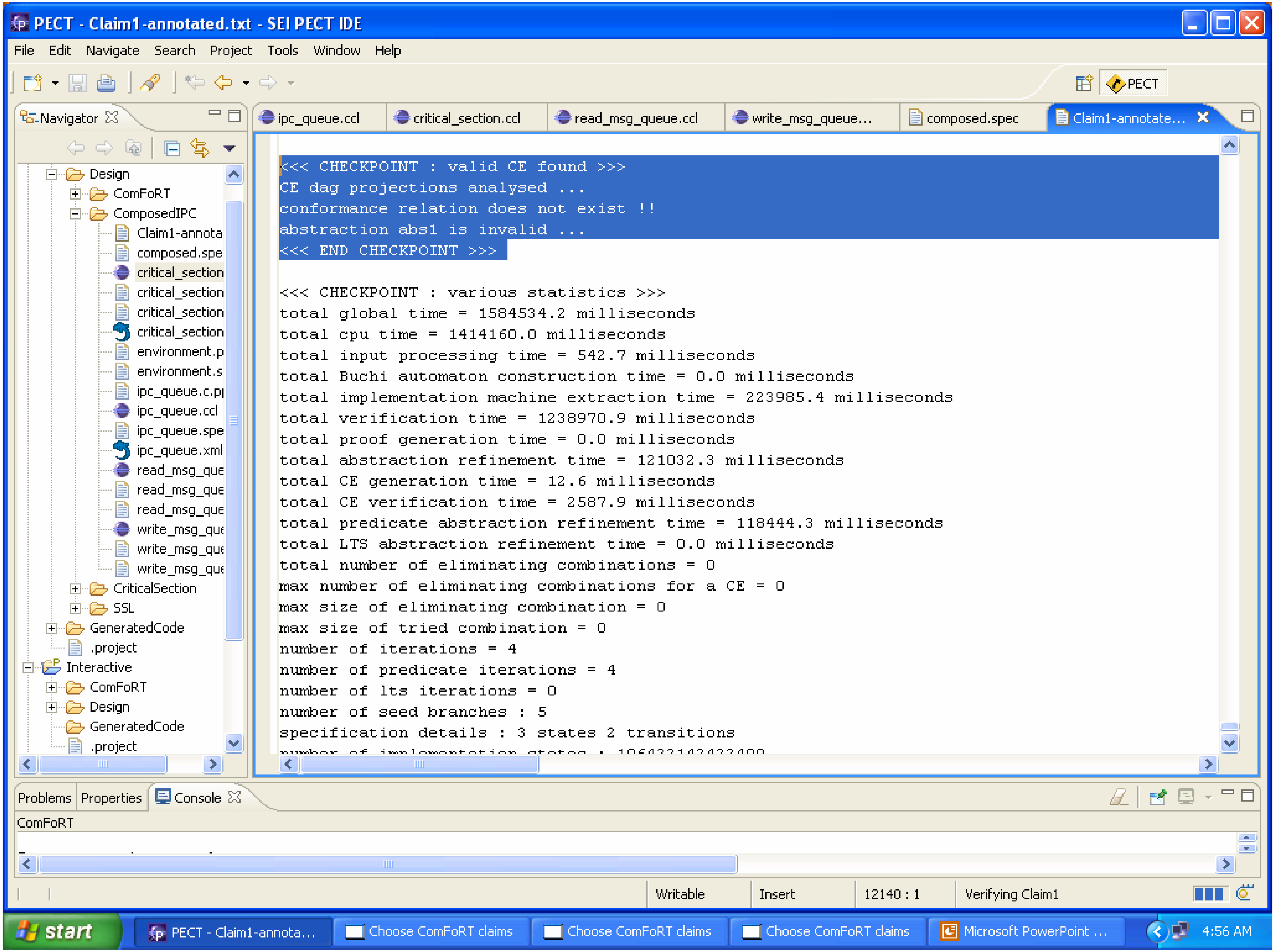

PECT - critical_section.ccl - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
Design
ComFoRT
ComposedIPC
Claim1-annotated.txt
composed.spec
critical_section.ccl
critical_section.pp
critical_section.spec
environment.pp
environment.spec
ipc_queue.c.pp
ipc_queue.ccl
ipc_queue.spec
ipc_queue.xml
read_msg_queue.ccl
read_msg_queue.pp
read_msg_queue.spec
write_msg_queue.ccl
write_msg_queue.pp
write_msg_queue.spec
CriticalSection
SSL
GeneratedCode
.project
Interactive
ComFoRT
Design
ipc_queue.ccl
critical_section.ccl
read_msg_queue.ccl
write_msg_queue.ccl
composed.spec
int numWaiting = 0;
int waiting1 = 0;
int type1 = 0;
int waiting2 = 0;
int type2 = 0;
int random = 0;
int error = 0;
int caller = 0;
sink mutex EnterCriticalSection_read (consume int caller);
sink mutex LeaveCriticalSection_read (consume int caller);
sink mutex EnterCriticalSection_write (consume int caller);
sink mutex LeaveCriticalSection_write (consume int caller);
threaded react CS (EnterCriticalSection_read, LeaveCriticalSection_read, EnterCr
start -> one {}
one -> two {trigger ^EnterCriticalSection_read(caller);}
two -> one {}
one -> three {trigger ^LeaveCriticalSection_read(caller);}
three -> one {}
one -> four {trigger ^EnterCriticalSection_write(caller);}
four -> one {}
one -> five {trigger ^LeaveCriticalSection_write(caller);}
five -> one {}
Problems Properties Console
ComFoRT
Verification complete.
Claim Claim28 does not hold. See targets_C1_Claim28.txt for the counterexample.
Demo/Design/ComposedIPC/critical_section.ccl
start
PECT - critical_sectio...
Microsoft PowerPoint ...
4:43 AM


PECT - critical_section.ccl - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
ipc_queue.ccl
critical_section.ccl
read_msg_queue.ccl
write_msg_queue.ccl
composed.spec
Design
ComFoRT
ComposedIPC
Claim1-annotated.txt
composed.spec
CriticalSecti
SSL
GeneratedCode
.project
Interactive
ComFoRT
Design
New
Open
Open With
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Generate C Code
Compile
Analyze Using
Interpretation
Properties
Performance RF
ComFoRT
int numWaiting = 0;
int waiting1 = 0;
int type1 = 0;
int waiting2 = 0;
int type2 = 0;
int random = 0;
int error = 0;
int caller = 0;
sink mutex EnterCriticalSection_read (consume int caller);
sink mutex LeaveCriticalSection_read (consume int caller);
sink mutex EnterCriticalSection_write (consume int caller);
sink mutex LeaveCriticalSection_write (consume int caller);
threaded react CS (EnterCriticalSection_read, LeaveCriticalSection_read, EnterCr
start -> one {}
one -> two {trigger ^EnterCriticalSection_read(caller);}
two -> one {}
one -> three {trigger ^LeaveCriticalSection_read(caller);}
three -> one {}
one -> four {trigger ^EnterCriticalSection_write(caller);}
four -> one {}
one -> five {trigger ^LeaveCriticalSection_write(caller);}
five -> one {}
Problems Properties Console
ComFoRT
Verification complete.
Claim Claim28 does not hold. See targets_C1_Claim28.txt for the counterexample.
Demo/Design/ComposedIPC/critical_section.ccl
start
PECT - critical_sectio...
Microsoft PowerPoint ...
4:44 AM


PECT - composed.spec - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
read_msg_queue.s...
critical_section.ccl
composed.spec
write_msg_queue....
ipc_queue.ccl
Claim1-annotated....
Demo
Design
ComFoRT
ComposedIPC
abstract abs7,{(P0::write_msg_queue_queue == 0) && (P0::write_msg_queue_msg == 0) ,
(P1::read_msg_queue_queue == 0) && (P1::read_msg_queue_src == 0) && (P1::read_msg_queue_ms
(P2::ipc_queue_writeTimeout == 0) && (P2::ipc_queue_readTimeout == 0) && (P2::ipc_queue_nu
(P2::ipc_queue_size == 2) && (P2::ipc_queue_message1 == 0) && (P2::ipc_queue_source1 == 0)
(P2::ipc_queue_message2 == 0) && (P2::ipc_queue_source2 == 0) && (P2::ipc_queue_error == 0
(P3::critical_section_owner == 0) && (P3::critical_section_ownerType == 0) && (P3::critica
(P3::critical_section_numWaiting == 0) && (P3::critical_section_waiting1 == 0) && (P3::cri
(P3::critical_section_waiting2 == 0) && (P3::critical_section_type2 == 0) &&
(P3::critical_section_random == 0) && (P3::critical_section_error == 0) && (P3::critical_s
}
//Claim0: G([P3::critical_section_error == 0]);
Claim0 = (epsilon -> ERROR0).
Claim0 = {}{P3::critical_section_error == 0};
//Claim1: G(begin_WaitForSingleObject_write => [P2::ipc_queue_numMessages == P2::ipc_queue_size]);
Claim1 = (epsilon -> S11).
= (begin_WaitForSingleObject_write -> ERROR1).
= {}{P2::ipc_queue_numMessages == P2::ipc_queue_size};
laim2: G([P2::ipc_queue_error == 0]);
im2 = (epsilon -> ERROR2).
im2 = {}{P2::ipc_queue_error == 0};
laim3: G([P3::critical_section_numWaiting < 2]);
im3 = (epsilon -> ERROR3).
Claim3 = {}{P3::critical_section_numWaiting < 2};
Cho...
Choose a claim to analyze:
Claim0 on ipc_queue
Claim1 on ipc_queue
Claim2 on ipc_queue
Claim3 on ipc_queue
OK
Problems Properties Console
ComFoRT
4 well-formed claims found.
Demo/Design/ComposedIPC/ipc_queue.ccl
start
ComFoRT
The ComFoRT_...
Control Panel
PECT - compose...
Choose ComFoR...
Choose ComFoR...
10:06 AM


PECT - Claim1-annotated.txt - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
Design
ComFoRT
ComposedIPC
Claim1-annota
composed.spe
critical_section
critical_section
critical_section
critical_section
environment.p
environment.s
ipc_queue.c.p
ipc_queue.ccl
ipc_queue.spe
ipc_queue.xml
read_msg_que
read_msg_que
read_msg_que
write_msg_que
write_msg_que
write_msg_que
CriticalSection
SSL
GeneratedCode
.project
Interactive
ipc_queue.ccl
critical_section.ccl
read_msg_queue.ccl
write_msg_queue...
composed.spec
Claim1-annotate...
############ P6::STUTTER ############
P6::temp_var_110 = do_environment (& P6::x , & P6::y , & P6::z) : STUTTER
############ P6::STUTTER ############
P6::temp_var_110 = do_environment (& P6::x , & P6::y , & P6::z) : STUTTER
############ P6::STUTTER ############
P6::temp_var_110 = do_environment (& P6::x , & P6::y , & P6::z)
****** end local CE dag #6 *****
CE dag projections analysed ...
<<< END CHECKPOINT >>>
<<< CHECKPOINT : Projection of CE on fourth component >>>
***** start local CE dag #3 ****
P3::curState = 145
############ {P3::curState = [$0 == 145]} ############
P3::critical_section_owner = 0
############ {P3::critical_section_owner = [$0 == 0]} ############
P3::critical_section_ownerType = 0
############ {P3::critical_section_ownerType = [$0 == 0]} ############
P3::critical_section_timesEntered = 0
############ {P3::critical_section_timesEntered = [$0 == 0]} ############
P3::critical_section_numWaiting = 0
############ {P3::critical_section_numWaiting = [$0 == 0]} ############
P3::critical_section_waiting1 = 0
############ {P3::critical_section_waiting1 = [$0 == 0]} ############
P3::critical_section_type1 = 0
############ {P3::critical_section_type1 = [$0 == 0]} ############
Problems Properties Console
ComFoRT
Pre-processing complete.
Starting verification as background task...
Writable
Insert
2945 : 19
Verifying Claim1
start
PECT - Claim1-annota...
Choose ComFoRT claims
Choose ComFoRT claims
Choose ComFoRT claims
Microsoft PowerPoint ...
4:50 AM


PECT - Claim1-annotated.txt - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
Design
ComFoRT
ComposedIPC
Claim1-annota
composed.spe
critical_section
critical_section
critical_section
critical_section
environment.p
environment.s
ipc_queue.c.p
ipc_queue.ccl
ipc_queue.spe
ipc_queue.xml
read_msg_que
read_msg_que
read_msg_que
write_msg_que
write_msg_que
write_msg_que
CriticalSection
SSL
GeneratedCode
.project
Interactive
ipc_queue.ccl
critical_section.ccl
read_msg_queue.ccl
write_msg_queue...
composed.spec
Claim1-annotate...
model extracted in 27638.8 milliseconds ...
model loaded from file ...
implementation states for control locations computed ...
model extracted in 443.5 milliseconds ...
model saved in 73.6 milliseconds ...
implementation states for control locations computed ...
model extracted in 27862.8 milliseconds ...
<<< CHECKPOINT : new predicates inferred from spurious counterexample >>>
Seeding with branch (P3::critical_section_owner == 0)
Seeding with branch (P3::critical_section_owner == P3::critical_section_caller)
Seeding with branch (P3::critical_section_owner == 0)
Seeding with branch (P3::critical_section_owner == P3::critical_section_caller)
<<< END CHECKPOINT >>>
spurious counter-example detected !!
abstraction refined !!
starting iteration number 3 ...
model saved in 75.7 milliseconds ...
number of abstract implementation states = 3975
implementation states for control locations computed ...
action-guided transitions computed ...
model extracted in 25843.6 milliseconds ...
implementation machine extracted in 25844.2 milliseconds ...
global states : (280 284 626 3975 7 7 2) = 19391612016000
Problems
Properties
Console
ComFoRT
Pre-processing complete.
Starting verification as background task...
Writable
Insert
6344 : 31
Verifying Claim1
start
PECT - Claim1-annota...
Choose ComFoRT claims
Choose ComFoRT claims
Choose ComFoRT claims
Microsoft PowerPoint ...
4:53 AM


PECT - Claim1-annotated.txt - SEI PECT IDE
File Edit Navigate Search Project Tools Window Help
PECT
Navigator
ipc_queue.ccl
critical_section.ccl
read_msg_queue.ccl
write_msg_queue...
composed.spec
Claim1-annotate...
Design
ComFoRT
ComposedIPC
Claim1-annota
composed.spe
critical_section
critical_section
critical_section
critical_section
environment.p
environment.s
ipc_queue.c.p
ipc_queue.ccl
ipc_queue.spe
ipc_queue.xml
read_msg_que
read_msg_que
read_msg_que
write_msg_que
write_msg_que
write_msg_que
CriticalSection
SSL
GeneratedCode
.project
Interactive
ComFoRT
Design
GeneratedCode
.project
<<< CHECKPOINT : valid CE found >>>
CE dag projections analysed ...
conformance relation does not exist !!
abstraction abs1 is invalid ...
<<< END CHECKPOINT >>>
<<< CHECKPOINT : various statistics >>>
total global time = 1584534.2 milliseconds
total cpu time = 1414160.0 milliseconds
total input processing time = 542.7 milliseconds
total Buchi automaton construction time = 0.0 milliseconds
total implementation machine extraction time = 223985.4 milliseconds
total verification time = 1238970.9 milliseconds
total proof generation time = 0.0 milliseconds
total abstraction refinement time = 121032.3 milliseconds
total CE generation time = 12.6 milliseconds
total CE verification time = 2587.9 milliseconds
total predicate abstraction refinement time = 118444.3 milliseconds
total LTS abstraction refinement time = 0.0 milliseconds
total number of eliminating combinations = 0
max number of eliminating combinations for a CE = 0
max size of eliminating combination = 0
max size of tried combination = 0
number of iterations = 4
number of predicate iterations = 4
number of lts iterations = 0
number of seed branches : 5
specification details : 3 states 2 transitions
Problems Properties Console
ComFoRT
Writable
Insert
12140 : 1
Verifying Claim1
start
PECT - Claim1-annota...
Choose ComFoRT claims
Choose ComFoRT claims
Choose ComFoRT claims
Microsoft PowerPoint ...
4:56 AM

Applications

IPC Module

- Deployed by a world leader in robotics
- Discovered synchronization bug under which senders would receive the wrong answer to their requests
- Problem had remained undetected for seven years prior to independent discovery by business unit

Case Study: Micro-C OS

- Real-time OS for embedded applications
  - 6000+ LOC, widely used
- Verified locking discipline
- Found four bugs
  - Missing unlock and return
  - Three already reported

Ongoing and Future Work

- Use a SAT solver for computing abstraction
  - Semantics of bit-wise operators is taken into account
- Use of pattern languages for specifying properties
- Integrated Abstraction and Compositional reasoning
- Component certification

ComFoRT Resources

ComFoRT tools

- http://www.sei.cmu.edu/pacc/comfort.html

Ongoing industrial & academic collaborations

- Prof. Edmund Clarke and his model checking group, Prof. Peter Lee at CMU
- Prof. Dr. Daniel Kroening from ETH Zurich
- Industrial corporate research centers developing embedded controllers

Conference and Journal publications

References

Overview of ComFoRT: A Model Checking Reasoning Framework, CMU/SEI Tech. Report SEI-2004-TN-018 by James Ivers and Natasha Sharygina

State/Event-based Software Model Checking, In Proceedings of IFM *Integrated Formal Methods 2004 International Conference,* by Sagar Chaki, Edmund Clarke, Joel Ouaknine, Natasha Sharygina and Nishant Sinha.

Automated, compositional and iterative deadlock detection, In Proceedings of *the Second ACM-IEEE International Conference on Formal Methods for Codesign (MEMOCODE) 2004* , by Sagar Chaki, Edmund Clarke, Joel Ouaknine and Natasha Sharygina

Dynamic Component Substitutability Analysis, In Proceedings of FM 2005 *Formal Methods Conference, by Sagar Chaki,* Ed Clarke, Natasha Sharygina and Nishant Sinha.

References

Verification of Evolving Software, In Proceedings of SAVCBS 2004 by Sagar Chaki, Natasha Sharygina and Nishant Sinha

Snapshot of CCL: A Language for Predictable Assembly, In *CMU/SEI TR-2002-TR-031*, by James Ivers and Kurt Wallnau

A Technology for Predictable Assembly from Certifiable Components (PACC), In CMU/SEI-TR-2003-TR-009, by Kurt Wallnau

SAT-based predicate abstraction for ANSI-C, In *Formal Methods System Design* Journal, Vol. 25(2), 2004, by Daniel Kroening, Ed Clarke, Natasha Sharygina and Karen Yorav.