Shape Analysis

Mooly Sagiv

. . . and also

- University of Wisconsin
  - F. DiMaio
  - D. Gopan
  - A. Loginov
  - T. Reps
- IBM Research
  - J. Field
  - H. Kolodner
  - M. Rodeh
  - E. Yahav
- Microsoft Research
  - J. Berdine
  - B. Cook
  - G. Ramalingam
- University of Massachusetts
  - N. Immerman
  - B. Hesse
- Inria
  - B. Jeannet
- Tel-Aviv University
  - D. Amit
  - I. Bogudlov
  - G. Arnold
  - G. Erez
  - N. Dor
  - T. Lev-Ami
  - R. Manevich
  - R. Shaham
  - A. Rabinovich
  - N. Rinetzky
  - G. Yorsh
  - A. Warshavsky
- Universität des Saarlandes
  - J. Bauer
  - R. Biber
  - R. Wilhelm

Shape Analysis [Jones and Muchnick 1981]

- Determine the possible shapes of a dynamically allocated data structure at a given program point

Programs and Properties

- Dynamically allocated memory
- Recursive data structures
- Recursive procedures
- Concurrency

- Memory safety
- Preservation of Data structure invariants
- Partial correctness
- Termination
- Linearizability

Outline

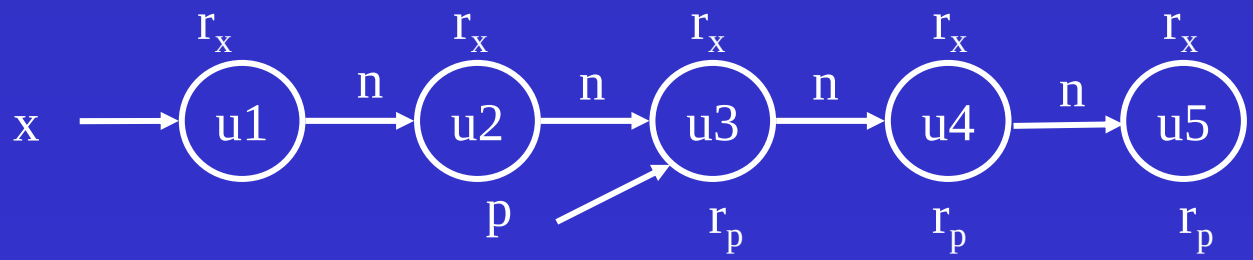
- Shape abstractions in a nutshell
- Computing transformers
- Heap decomposition

Representing Concrete Stores by Logical Structures

- Parametric vocabulary
- Heap
  - Locations ≈ Individuals
  - Program variables ≈ Unary relations
  - Fields ≈ Binary relations

Representing Concrete Stores by Logical Structures

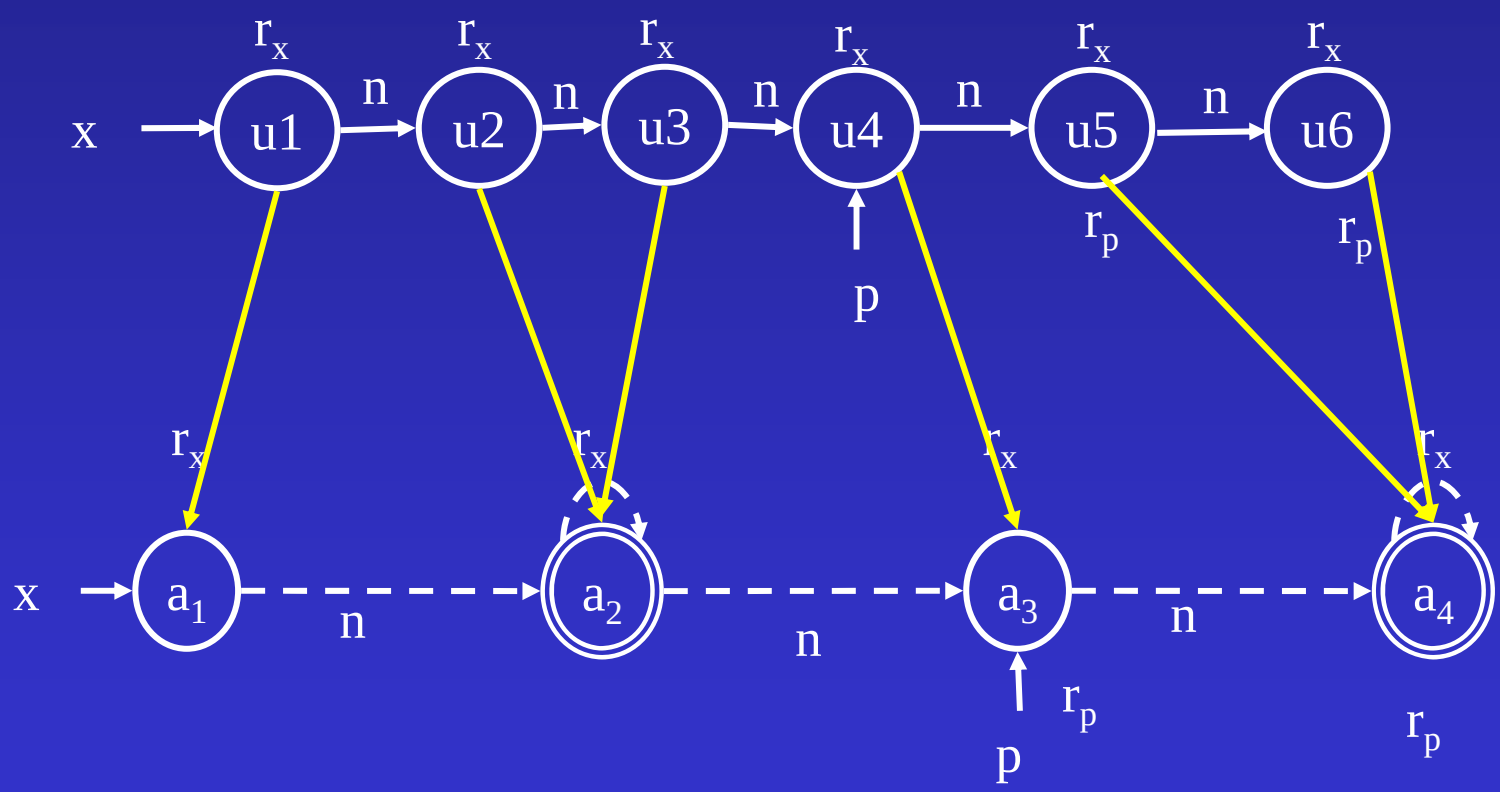
- U = {u1, u2, u3, u4, u5}
- x = {u1}, p = {u3}
- n = {<u1, u2>, <u2, u3>, <u3, u4>, <u4, u5>}
- r_x = {u1, u2, u3, u4, u5}
- r_p = {u3, u4, u5}



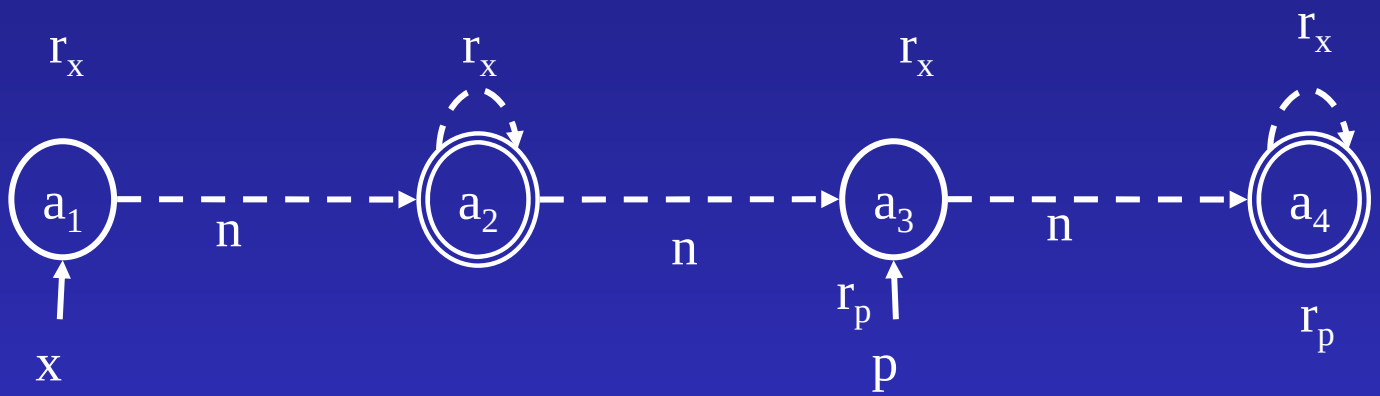
Representing Abstract Stores by 3-Valued Logical Structures

- A join semi-lattice: $0 \sqcup 1 = 1/2$
- {0, 1, ½} values for relations

Canonical Abstraction



Canonical Abstractions as Formulas [Yorsh'03, Kuncak'04, Wies'07]



$$\forall v: (x(v) \wedge r_x(v) \wedge \neg p(v) \wedge \neg r_p(v)) \vee$$
$$(\neg x(v) \wedge r_x(v) \wedge \neg p(x) \wedge \neg r_p(v)) \vee$$
$$(\neg x(v) \wedge r_x(v) \wedge p(v) \wedge r_p(v)) \vee$$
$$(\neg x(v) \wedge r_x(v) \wedge \neg p(v) \wedge r_p(v)))$$

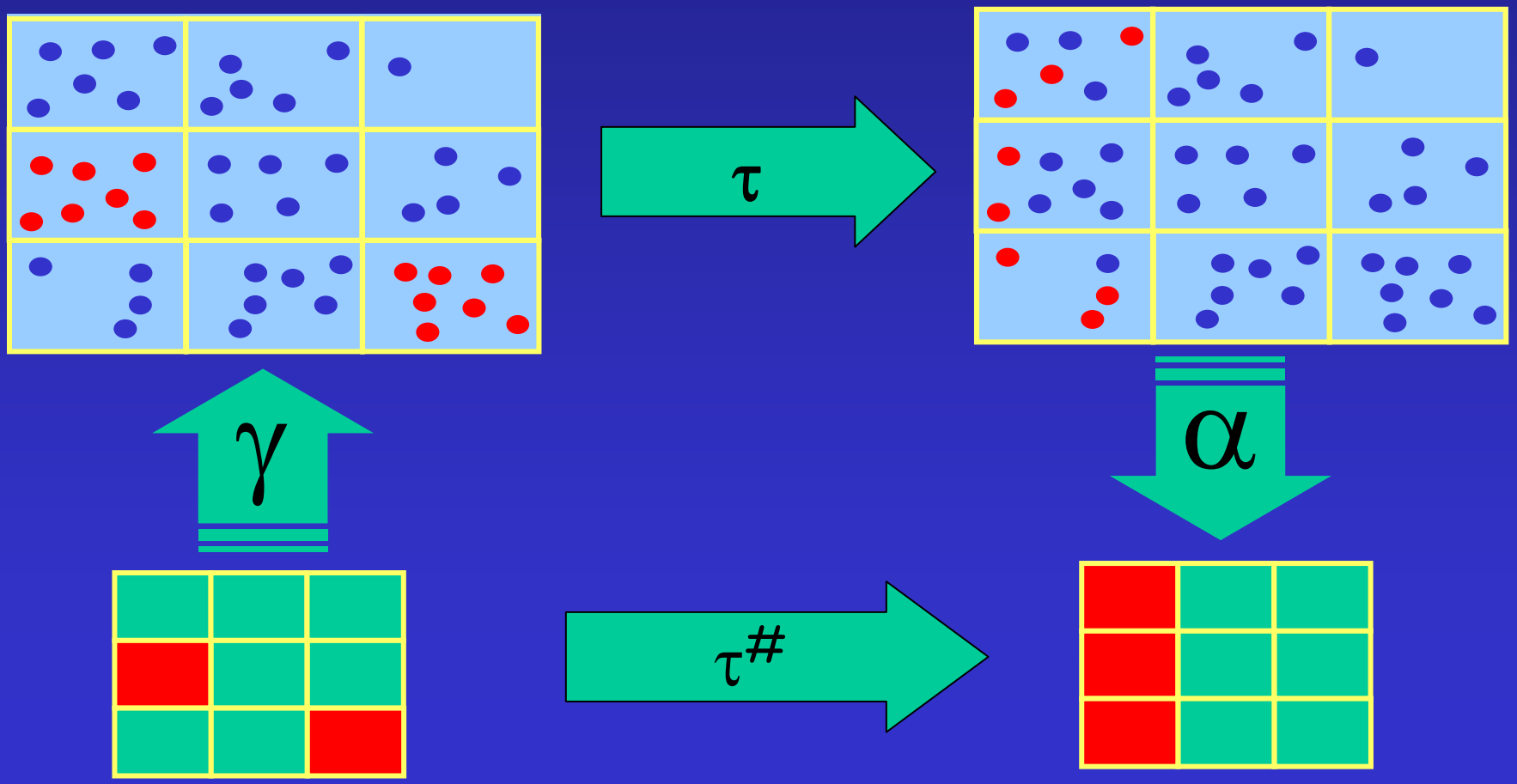
$$\forall v: r_x(v) \Leftrightarrow \exists w: x(w) \wedge n^*(w, v)$$

$$\forall v: r_p(v) \Leftrightarrow \exists w: p(w) \wedge n^*(w, v)$$

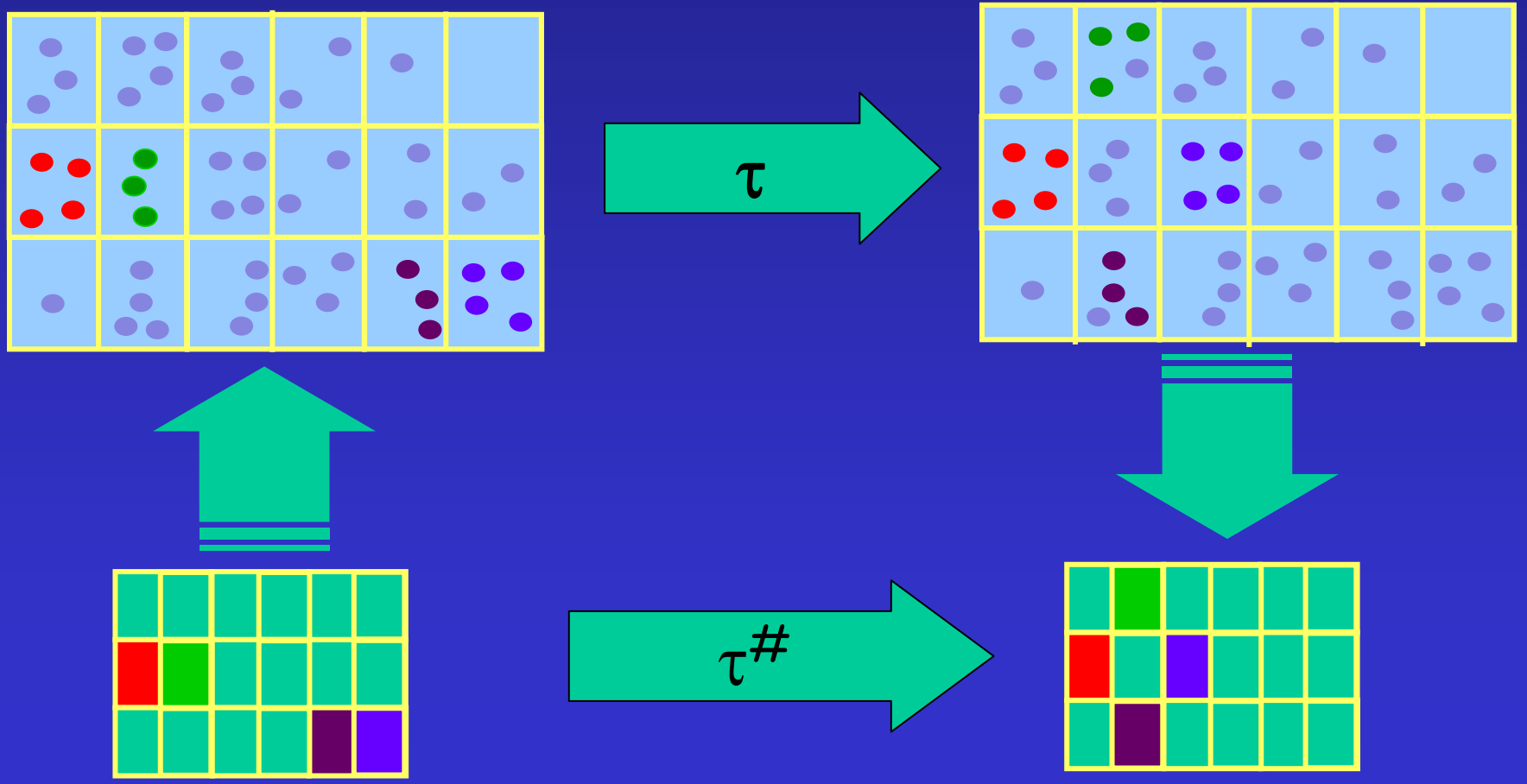
Canonical Abstraction

- Limited form of quantified invariants
  - quantifier alternation only in instrumentation
- Not a static memory partition
  - The same memory location can be represented by different abstract nodes in different shape graphs

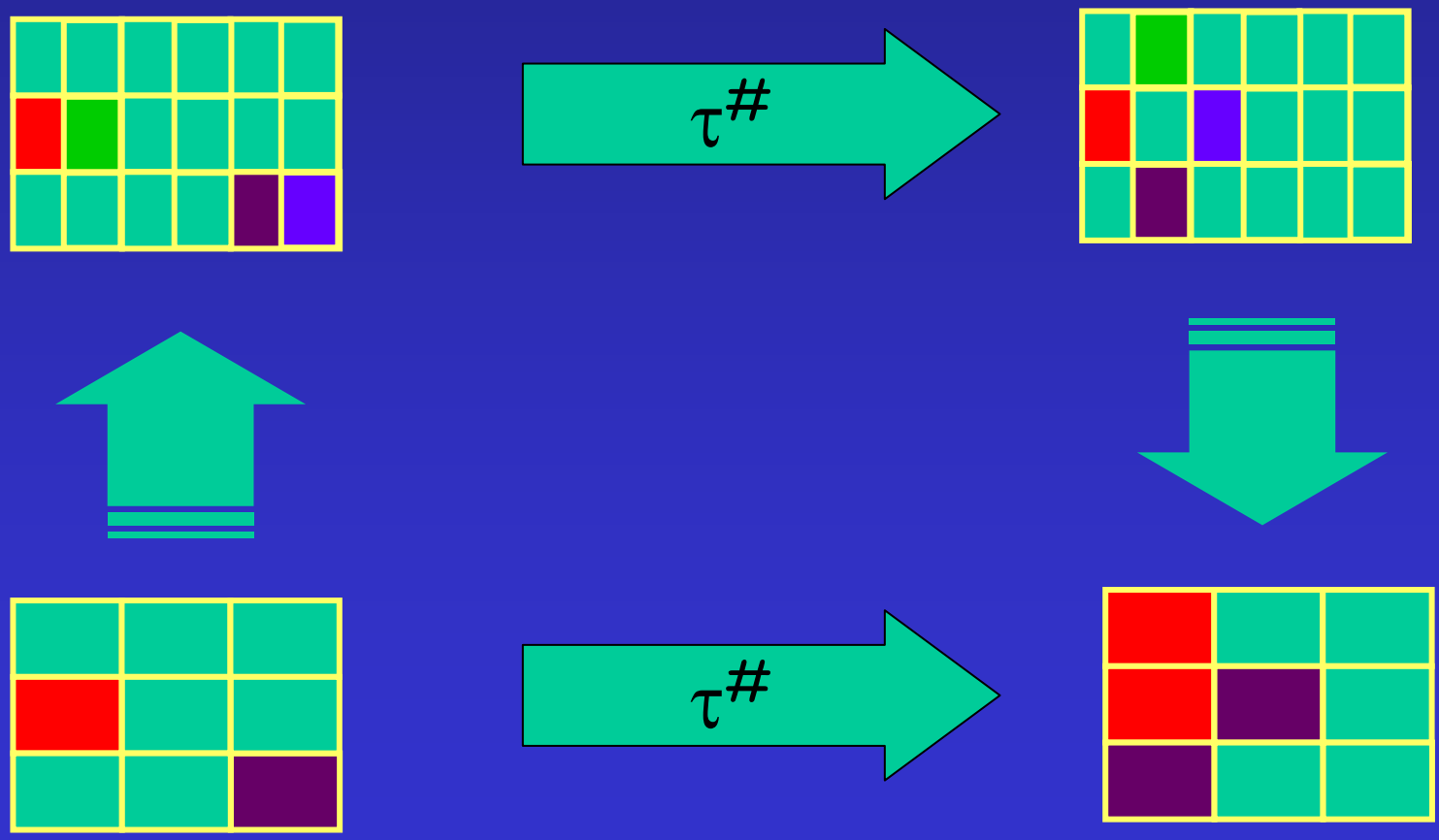
Most Precise Abstract Transformer [Cousot, Cousot POPL 1979]



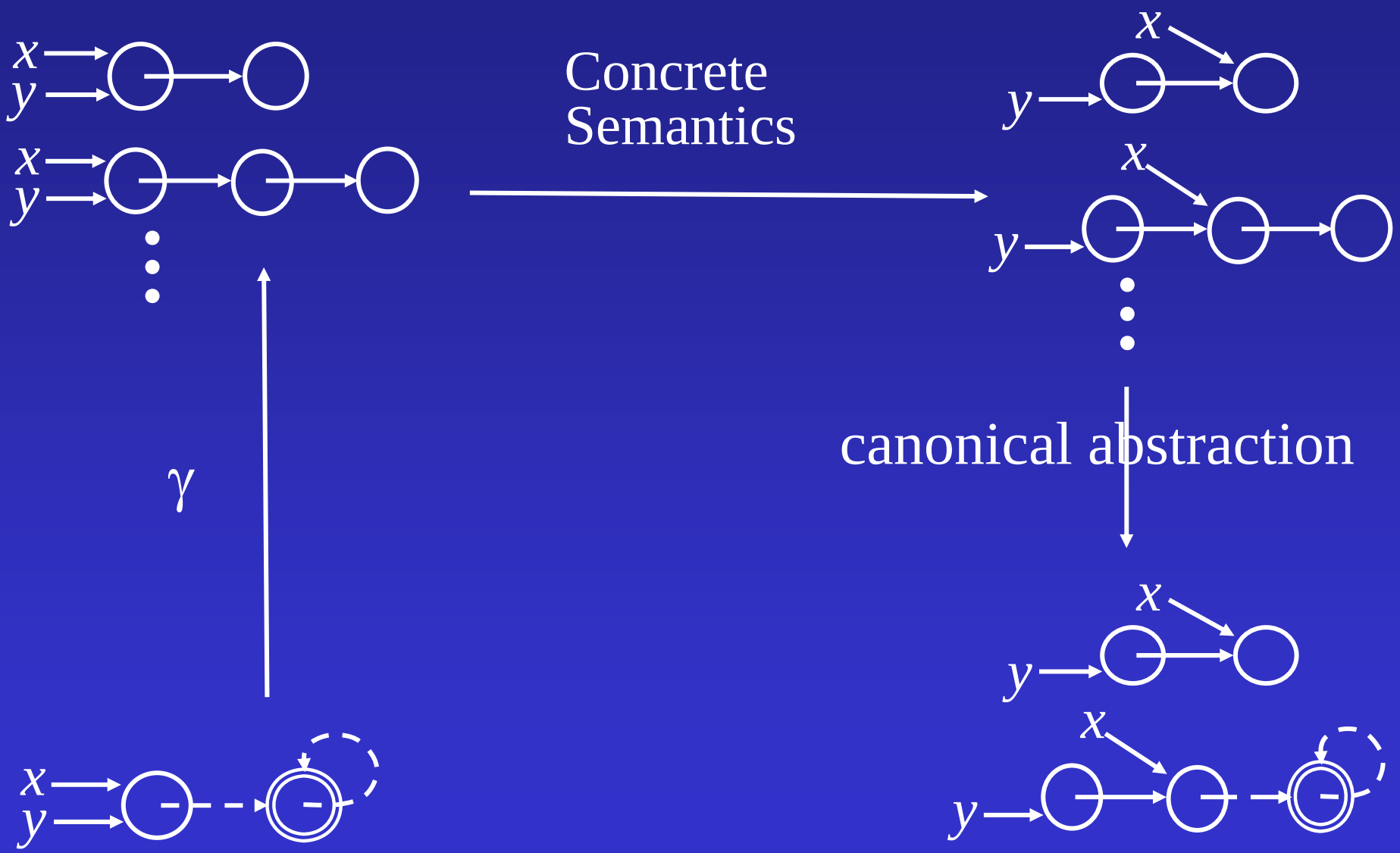
Partial Concretization



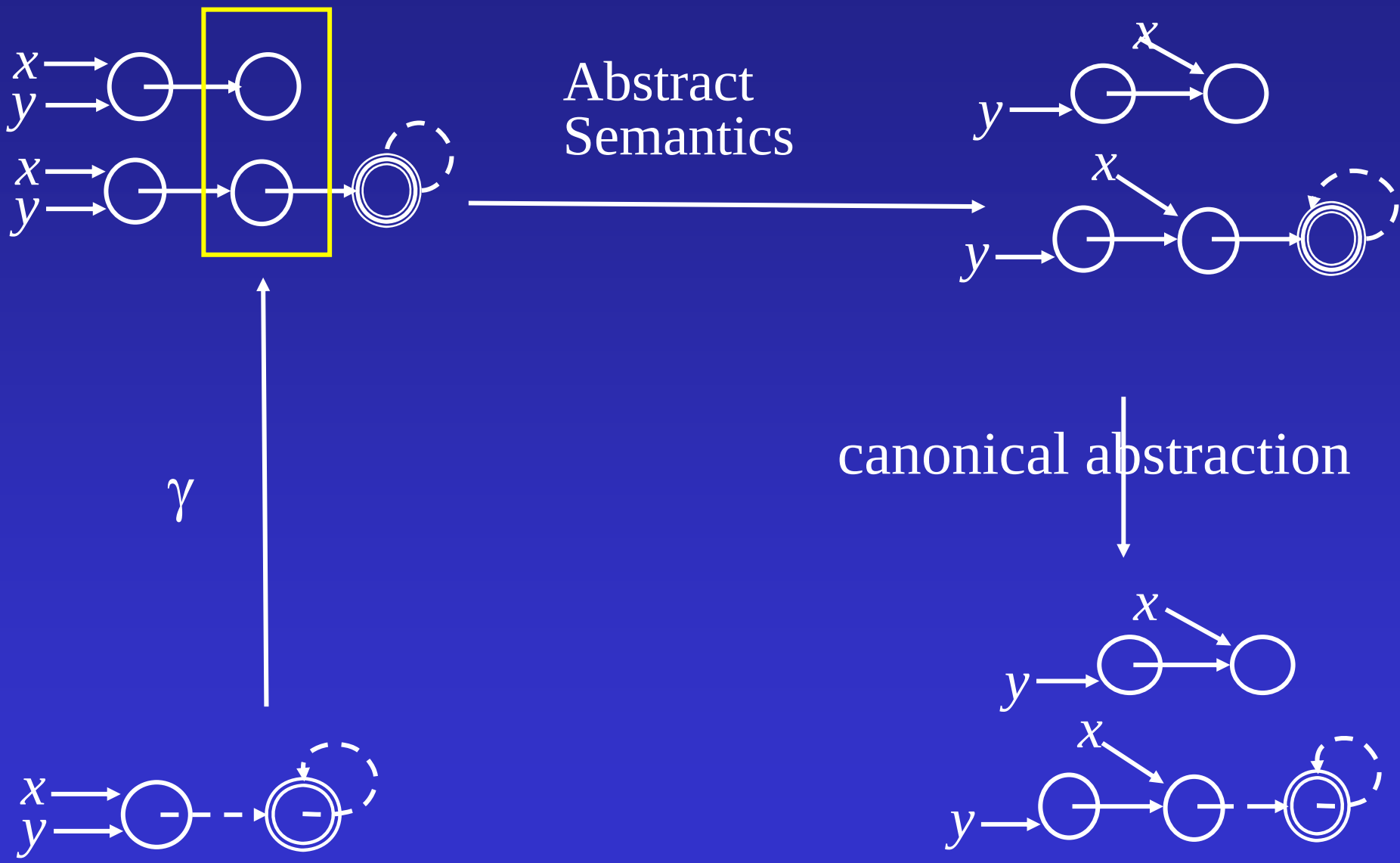
Partial Concretization



Best Transformer ($x = x \rightarrow n$)



Partial Concretization based Transformer ($x = x \rightarrow n$)



Partial Concretization

- Employed in other shape analysis algorithms [Distefano, TACAS'06, Evan, SAS'07, POPL'08]
- Soundness is immediate
- Can even guarantee precision under certain conditions [Lev-Ami, VMCAI'07]
- Locally refine the abstract domain per statement

Heap Decomposition for Concurrent Shape Analysis

Joint work with

R. Manevich
T. Lev-Ami
Tel Aviv University

G. Ramalingam
MSR India

J. Berdine
MSR Cambridge

Main Results

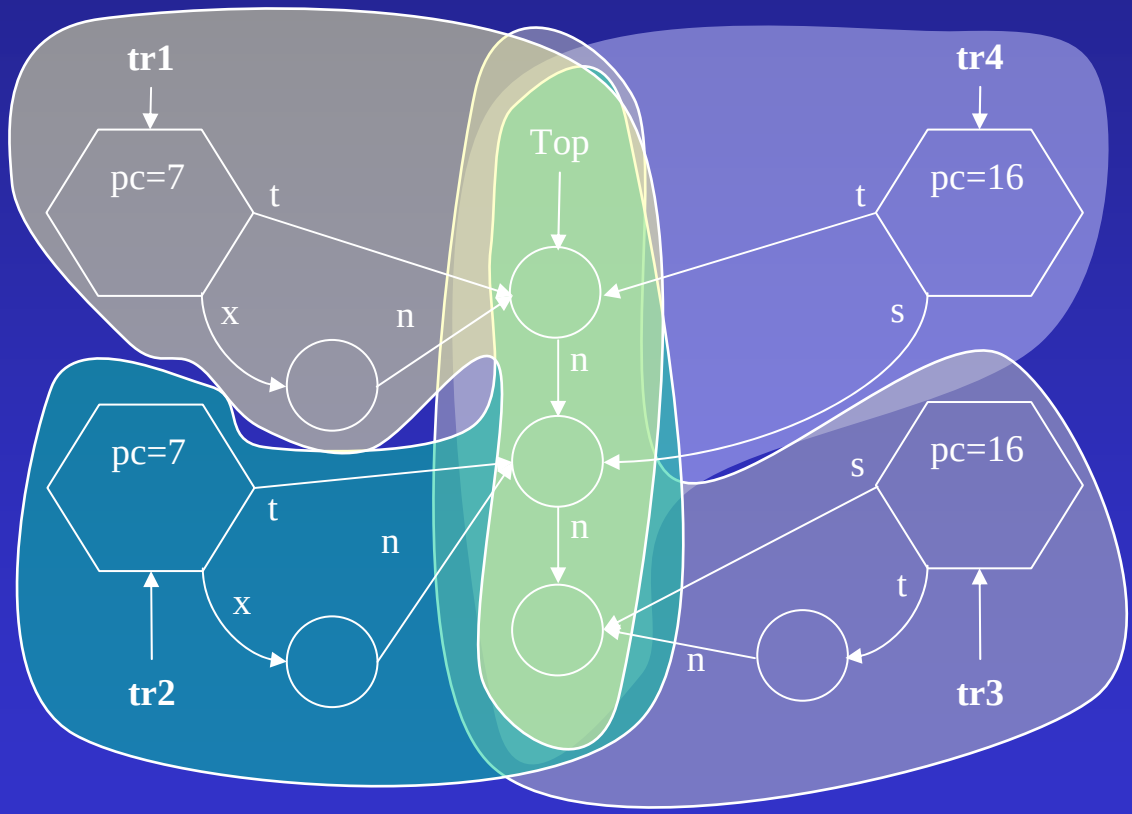
- New parametric abstraction for heaps
  - Heap decomposition + Cartesian product
- Exponential state space reduction
- Implementation in HeDec (Generalizes TVLA)
  - Heap Decomposition + Canonical abstraction
- Used to prove interesting properties of heap-manipulating programs with fine-grained parallelism
  - Linearizability

Treiber’s Non-blocking Stack

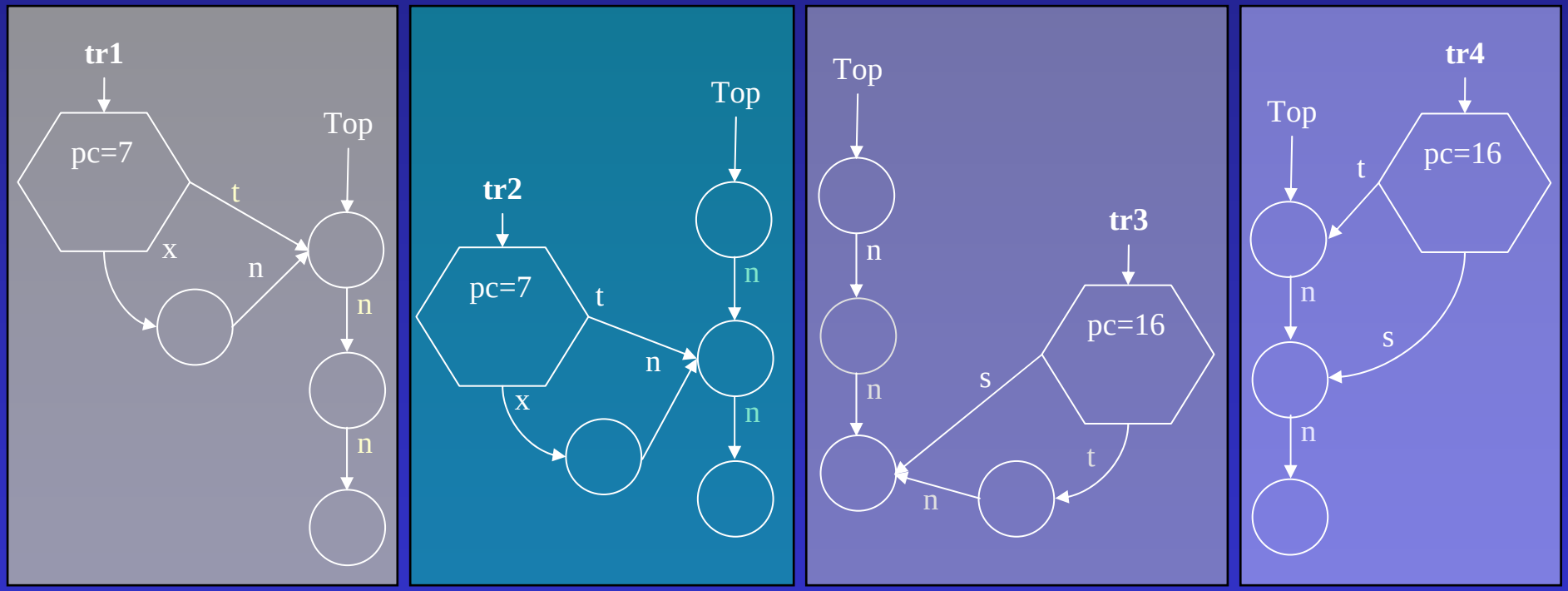
```
[1] void push(Stack *S, data_type v) {
[2]     Node *x = alloc(sizeof(Node));
[3]     x->d = v;
[4]     do {
[5]         Node *t = S->Top;
[6]         x->n = t;
[7]     } while (!CAS(&S->Top,t,x));
[8] }


[9]  data_type pop(Stack *S){
[10]     do {
[11]         Node *t = S->Top;
[12]         if (t == NULL)
[13]             return EMPTY;
[14]         Node *s = t->n;
[15]         data_type r = s->d;
[16]         } while (!CAS(&S->Top,t,s));
[17]     return r;
[18] }
```

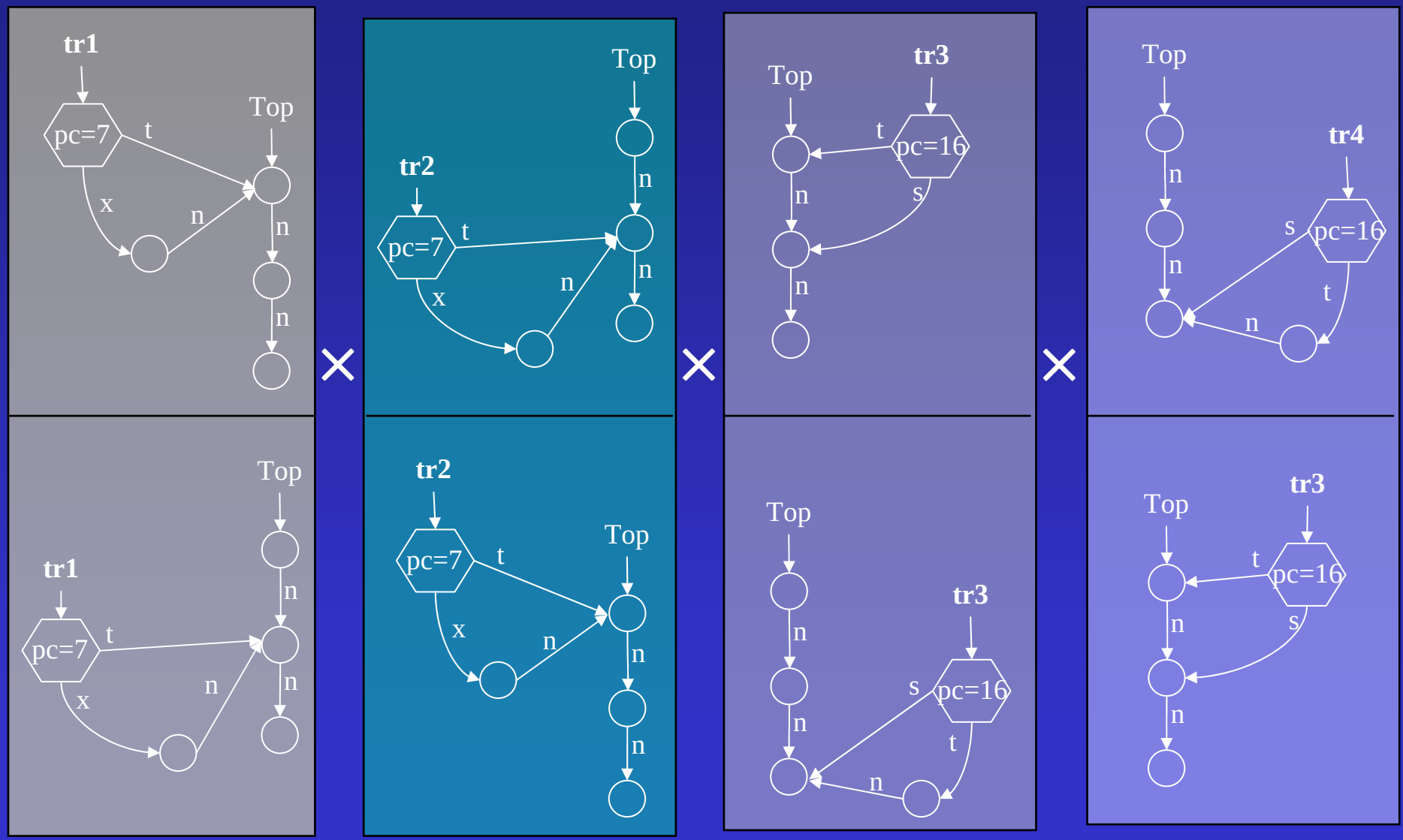
Full State



Sub-states

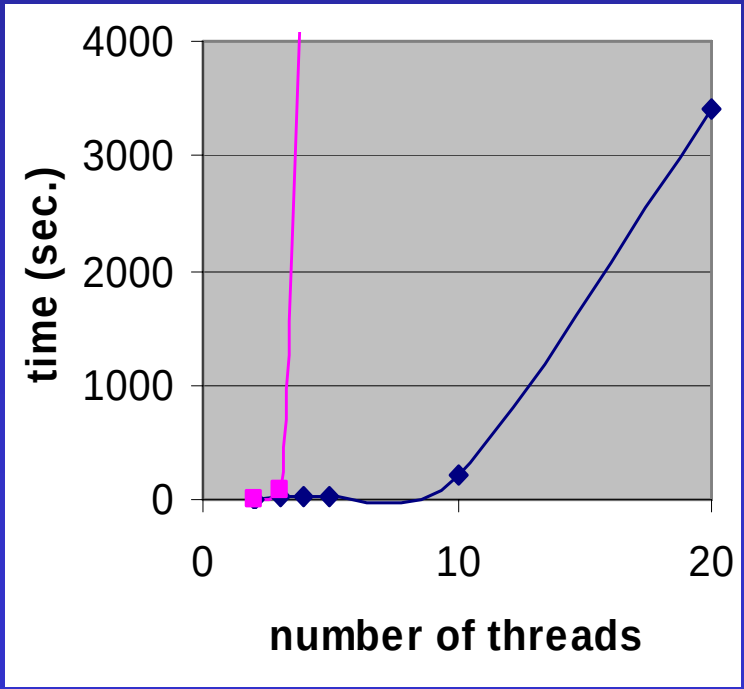

tr1
pc=7
Top
t
x
n
n
n
tr2
Top
pc=7
t
n
x
n
n
Top
n
tr3
pc=16
n
s
n
t
n
tr4
Top
pc=16
t
n
s
n

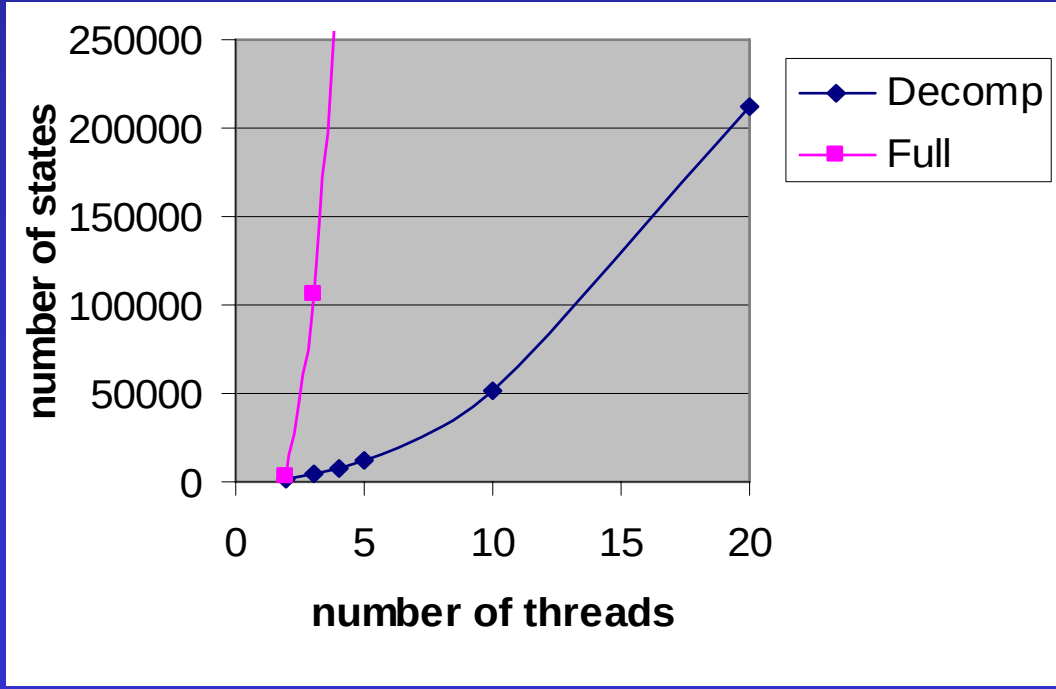
Cartesian Product of Sub-states



Empirical Results

- Exponential time/space reduction
  - Non-blocking stack + linearizability





and…

More information from
http://www.cs.tau.ac.il/~rumster

Thank you Cousot for

- Establishing the right mindset
- Galois Connections
- Semantic reductions
- Domain constructors

Summary

- Shape analysis is an interesting abstract interpretation problem
  - Handles unbounded memory
  - Partially disjunctive abstractions
- Partial concretization is useful for transformers
- Heap decomposition is useful for scalability
  - Generalizes thread-modular analysis
- Limited forms of quantified invariants can be utilized to prove interesting properties