Constraints for (Parameterized) Verification

Giorgio Delzanno
DIBRIS, UNIGE

CP2CAV

ITAP, June 28, 2012

Model Checking

- Ingredients of Model Checking
  - Operations on sets of states:

    union, intersection, etc

    pre/post operator:

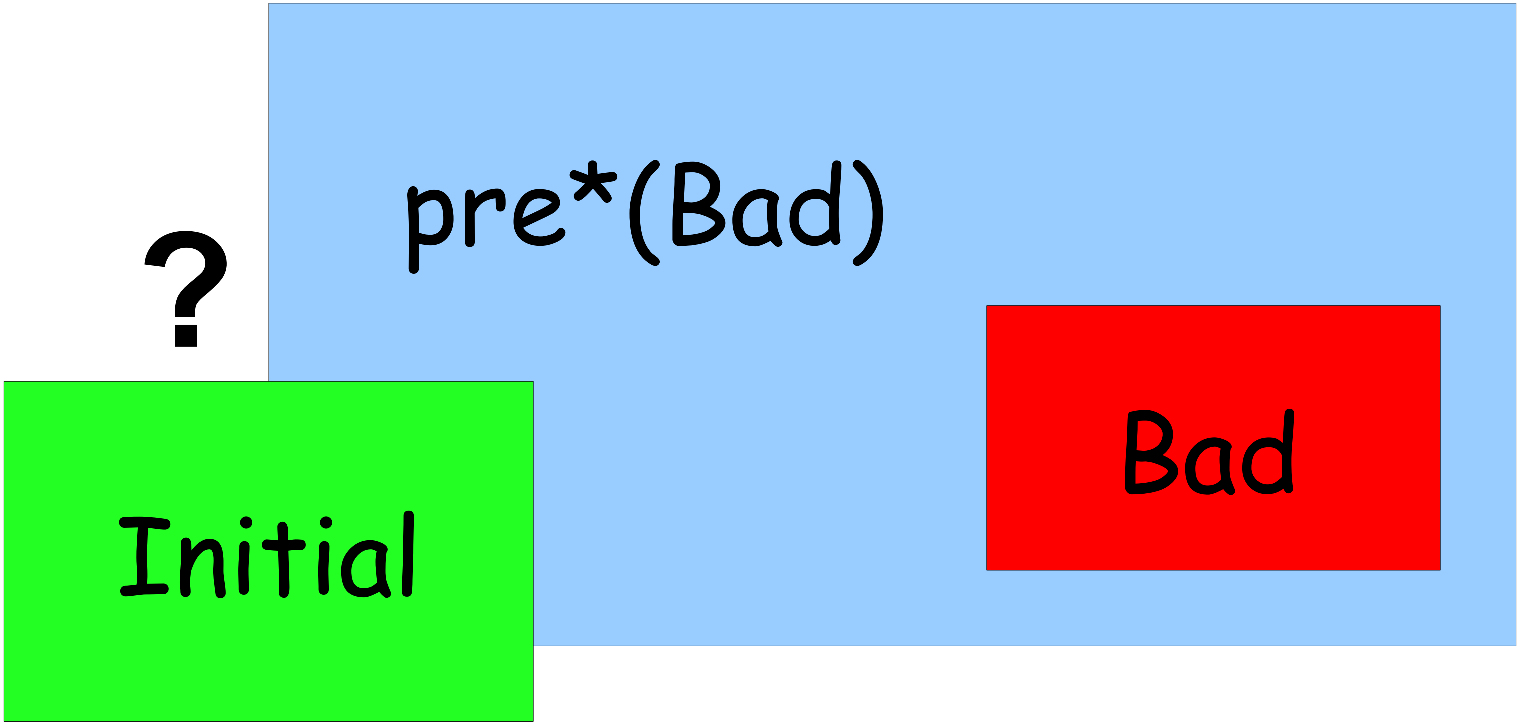
    transformation of sets of states
  - Fixpoint computation:

    pre* and post*

    (transitive closure of transition relation)

Safety: General Framework

- Verification of safety properties can be reduced to reachability of bad states



Symbolic Model Checking

- Ingredients of Symbolic Model Checking
  - Symbolic operations on sets of states:
    union, intersection, etc
  - pre/post operations:
    Symbolic transformation of sets of states
  - Fixpoint computation:
    pre* and post* symbolic computations

Symbolic Model Checking

- For finite state systems:
  - BDDs/Boolean Formulas
- For reachability in infinite-state systems
  - Regions of timed automata/Zones [Alur-Dill, Abdulla et al.]
  - Finite-state automata for pushdown systems [Bouajjani-Esparza-Maler,...] :
  - ...

Constraints 4 Safety

- Metaphor of constraints to generalize the role of BDDs in symbolic model checking
- General requirements to obtain effective and terminating procedures to check reachability
- Focus on constraints for systems composed of an arbitrary but finite number of component (Parameterized Verification)

[Fribourg,Delzanno-Podelski,Abdulla-Jonsson,...]

Constraints 4 Safety

Consider a system with set of states Q,

A constraint system $(C,<)$ is such that

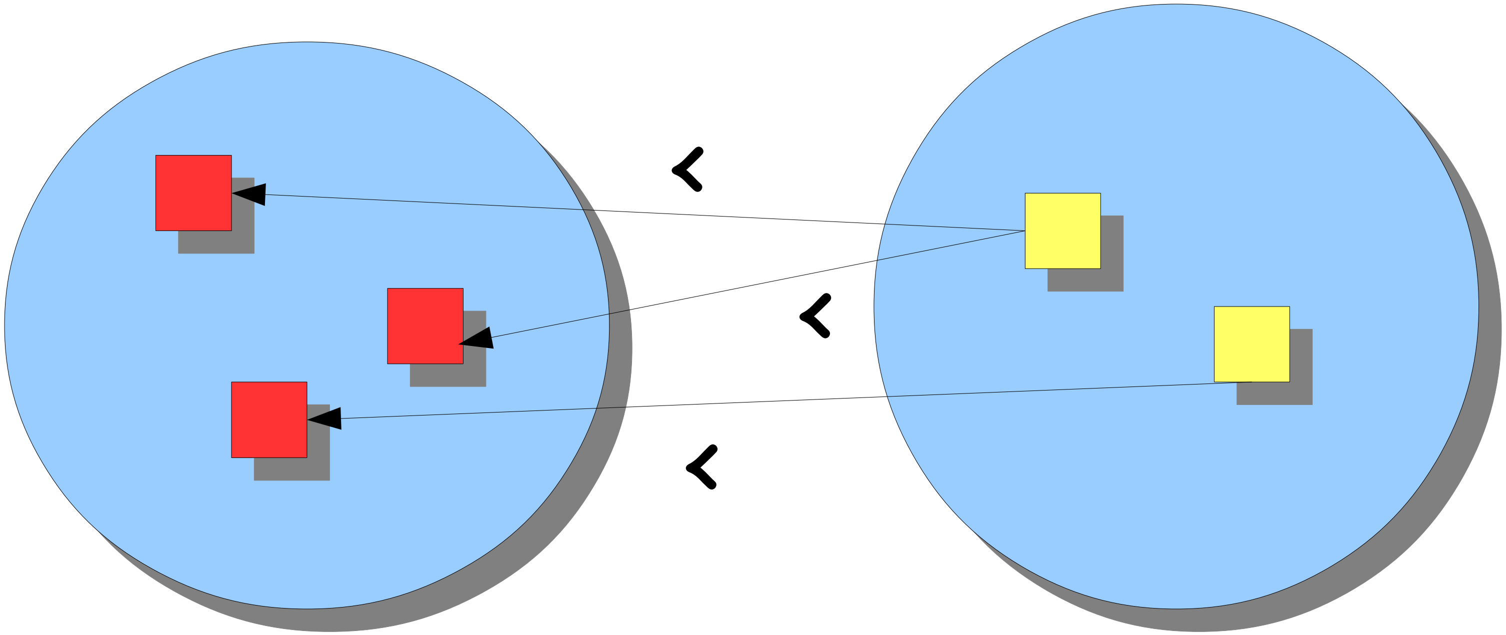
(denotation) [c] is a subset of Q for c in C

(entailment) $c<d$ implies [d] is contained into [c]

[Abdulla-Cerans-Jonsson-Tsay,Abdulla-Jonsson]

Finite sets of constraints

S << S'



S << S' iff for each d in S' there exists c in S s.t. c<d

Ingredients for Reachability

- Representation of initial/bad states (I and B)
- Decidable entailment test ‹
- Algorithm for computing predecessors, i.e.,

  Pre(S)=S' s.t. [S']=pre([S])
- Decidable test "I intersects S"

Naive Backward Reachability

1. R:=B; (set of constraints x bad states)
2. O:=R; (to check stability)
3. R:=(R union Pre(R));
4. If (O << R) return (I intersects R)
5. goto 1.

:

Some Optimizations

- If d in Pre(c) first check if c < d then compare with the remaining constraints in Old
- Eliminate redundant constraints in O, i.e., all constraints S that are subsumed by new constraints (and constraints that generated S)
- Specific strategies for computing Pre (e.g. always try to compute first the "more" general constraints perhaps using a mix dfs and bfs)
- ...

Ensuring Termination

- $(C,<)$ is a well quasi ordering (wqo) iff for every sequence of constraints c1 c2 c3 … there exist $i < j$ s.t. $c_i < c_j$
- If $(C,<)$ is a wqo, then every chain S_1 S_2 … (e.g. sets computed during backward reachability) eventually stabilizes, i.e., there exists k s.t. $[S_k]=[S_{k+1}]$

Note: it only works for chains

Petri Nets

- k counters, ++, --, no zero-test
  - Configurations are vectors of natural numbers
  - m<m' if m(p) is less or equal than m'(p) for every place p (wqo by Dickson's lemma)
- We can solve coverability:
  - Given m and m', starting from m can we reach m'' s.t. m'<m''?

Termination is guaranteed!

The Role of Constraint Solvers

- Constraint solvers can be used as engine for:
  - Computing Pre (renaming/projection)
  - Check entailment
  - Check intersection with initial states
- Observation:
  - In general we need sets of constraints (disjunctions) we may work with approximations
  - Termination guarantees vs practical termination

Some examples

- ALV: BDD+Omega [UCSB]
- CLP-based model checking: clp(R) [MPI]
- HyTech, PHaver: Polyhedra Lib, Parma Polyhedra Lib (PPL)
- Sharing Trees, Interval Sharing Trees [ULB]
- Combined representations TREX [Liafa]
- Automata for queues and integers [Liege]
- ....

Parameterized Verification

(coined by Sistla?)

Parameterized Verification

- Goal: verify safety for systems composed of an arbitrary but finite number of components
  - Petri nets (abstractions of multithreaded programs)
  - Broadcast protocols (abstractions of cache coherence protocols)
  - Skeletons of concurrent/distributed algorithms
  - Network protocols with different topologies (flooding, routing)

Several Approaches

- Invariants and theorem proving
- Abstractions and finite models (cut off points)
- Regular model checking
  (sets of configurations=automata)
- Forward/backward reachability

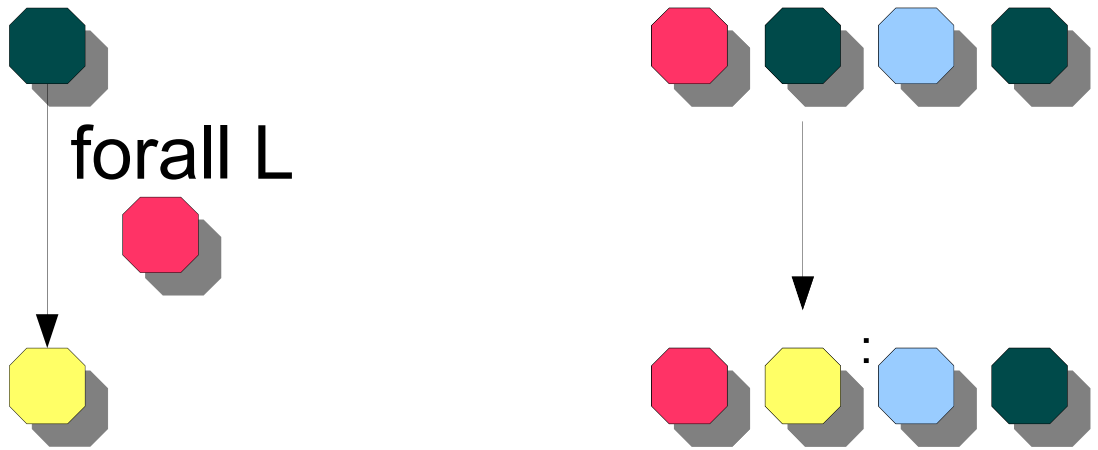
Focus: constraint-based approach

Linearly Ordered Systems

- A system is defined as an (unbounded) array of processes
- Each process is defined by an automata with guards
- Guards have the form:
  - Exists a process to the left/right with state q
  - All processes to the left/right have states in S
- We also consider rendez-vous and broadcast
- Examples: mutex and coherence protocols with atomic guards

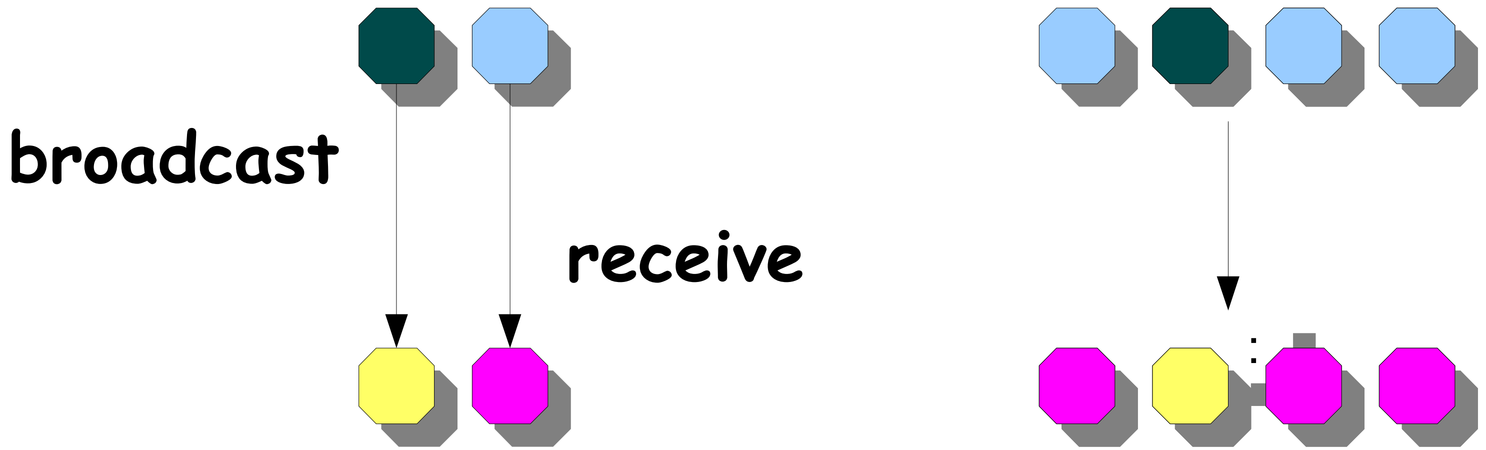
Semantics

- Configurations: words over a finite alphabet
- Transitions: word2word transformation
- E.g. with 4 processes



Semantics

- Configurations: words over a finite alphabet
- Transitions: word2word transformation
- E.g. with 4 processes



Safety

- Mutual exclusion: Reachable configuration contains at most one occurrence of state q (=critical section)
- Bad states: all configurations with two or more occurrences of state q
- Bad states are upward closed w.r.t. subword ordering (they are generated by the word qq) As a regular expression: $Q^*qQ^*qQ^*$

Analysis

- The use of universal quantification makes the model Turing complete
- Precise analysis --> no termination: automata/regular expressions as symbolic representations of sets of states
- Approximate analysis --> termination: upward closed sets of words

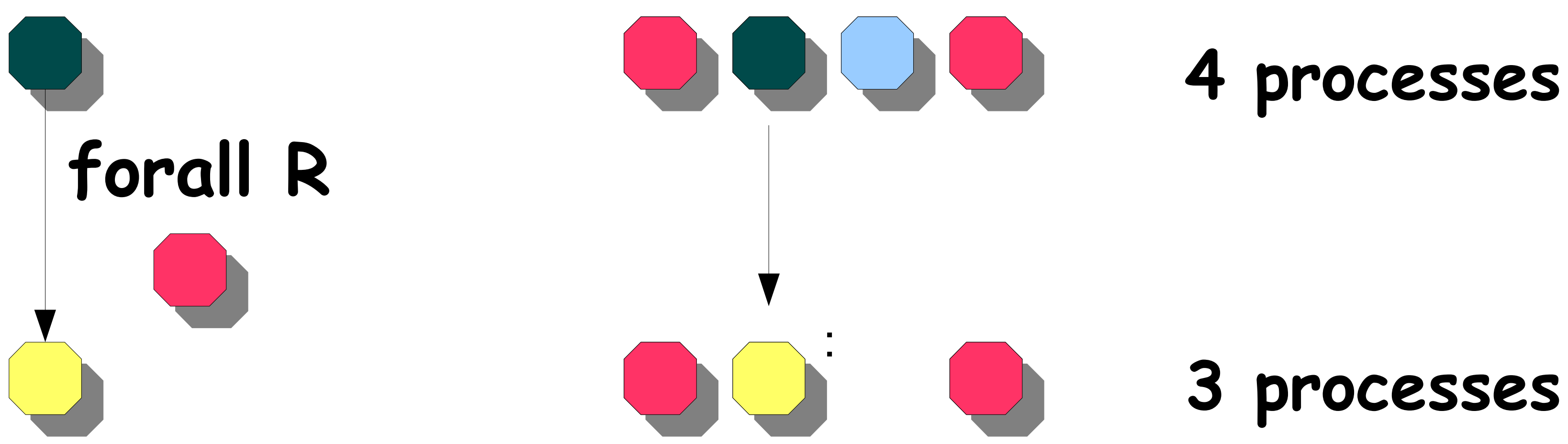
A Simple Abstraction

- We work with upward closed sets of words
- Finitely generated because subword is a wqo
- Constraint: a set of words B
  [B]= {w' | w subword of w', w in B}
- Approximation:
  Starting from an upward closed set S we compute the minimal upward closed set that contains pre(S)

[Monotonic Abstraction: Abdulla, Rezine,....]

A Simple Abstraction

- Operationally the abstraction corresponds to cancellation of processes that do not satisfy the guard:



Properties

- Backward reachability is guaranteed to terminate (subword is a wqo)
- Simple but it verifies safety on most of the examples of linearly ordered systems with atomic guards in the literature of parameterized systems

:

Spurious Traces

- It fails on
  - Ordered systems where processes in certain states act as sentinels (e.g. Szymanski's alg.)
  - Unordered systems (counter systems) in which there are variables that keep track of processes (e.g. readers/writers)
- In this examples upward closed sets are too rough

Patch for Ordered Systems

- We need to keep information coming from universal quantification
- We use r.e. (a1…an,P) = P* a1 P* … P*an P*
  P is contained in Q, a1,…,an in P
- New wqo: (w,P) < (w',P') iff
  w subword of w', P' is contained into P
- Idea: when computing Pre for a rule with guard forall S, (w,P) → (w', P intersect S)

[Delzanno-Rezine]

Cegar for Unordered Systems

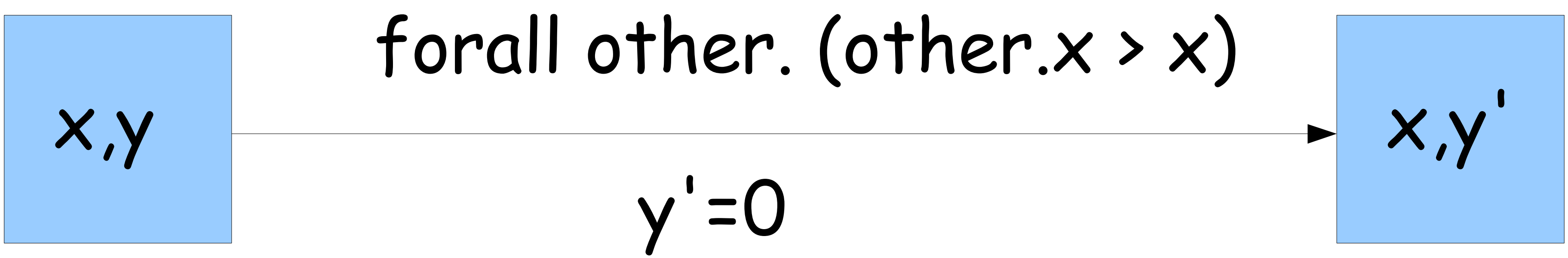
- For unordered systems we have defined an automatic refinement of the ordering
- The refinement computes interpolants for a pair of constraints in an abstract trace for which there exists no concrete transition connecting them
- Ordered case still open ...

Infinite-state processes

- Each process has a finite number of local variables ranging over integers
- Existential and universal global conditions where we compare variables of different processes
- Examples, e.g., Lamport's mutex (every process has local integer variables)
- Goal: try to verify mutual exclusion for any number of processes

Infinite-state processes

- Models = automata with data variables and global guards



:

Constraints?

We can use formulas over processes and relation over data

c = p(think,X),p(wait,Y),X<Y

Denotation: Upward closure w.r.t. multiset inclusion of all possible instances of p(think,X),p(wait,Y) obtained by taking solutions of X<Y

p(think,1),p(wait,2),p(think,2) belongs to [c]

Constraint solving?

- Satisfiability:
  we check satisfiability of numerical constraint

- Entailment:
  - injection of processes,
  - entailment of terms,
  - unification, projection and constraint entailment

Entailment

$c = p(S,X), p(wait,Y), X<Y$

$d = p(think,X'), p(wait,Y'), p(use,T'), X'<T', T'<Y'$

- S subsumes think
- $X<Y$ subsumes

  Exists $T'.\ X'<T', T'<Y', X=X', Y=Y'$
- [d] is contained into [c]

Non atomic Guards?

Non atomic guards are modelled via marking subprotocols (keep track of checked processes)

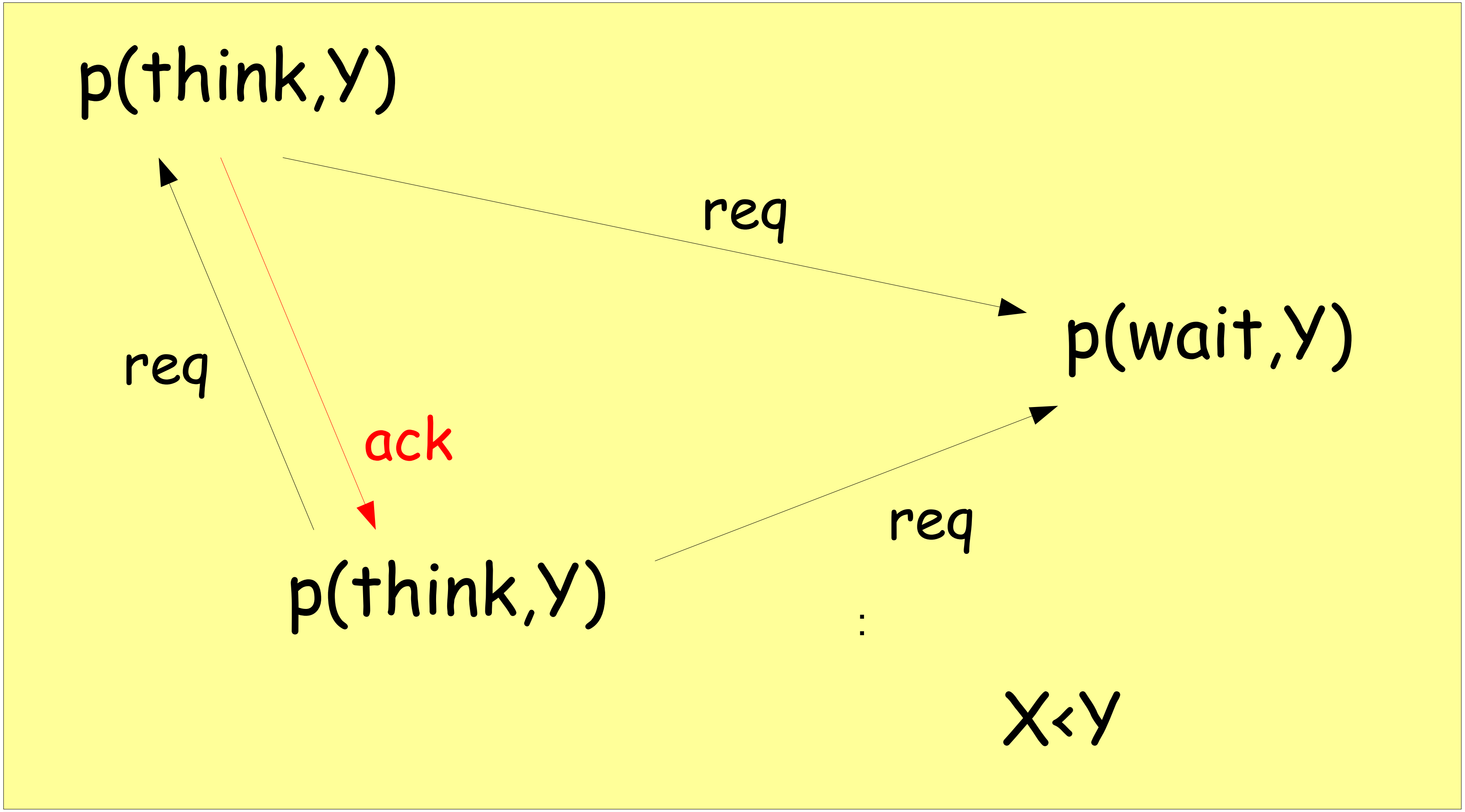
forall other. other.X > self.X

becomes

Pi: Send req(X) to every other process

Pj: Receive req(V) from Pi;
if X>V send(ack,Pi)

Constraints for non atomic guards = graphs



Abstractions and Termination

- We can still apply monotonic abstraction to work with upward closed sets (i.e. represent Pre as a finite sets of our constraints)
- Termination guarantees for special cases:
  - Guards are gap-order constraints
  - Each processes has at most 1 local variable

:

Gap order: x+c<y where c is a natural number

Implementation/results

- We have implemented ad hoc solvers in CLP(R) (to exploit unification and constraint solving) and PPL (Parma Polyhedra Lib) (to combine different symbolic representations like BDDs and constraints)
- We could verify safety for classical algorithms like Lamport's dist mutex, and Ricart Agrawala
- Non atomic Szymanski is still open

Other approaches

- Invariant checking with rich theories
- Forward + accelerations for counter systems and well-structured transition systems
- Static and dynamic cut-off points (try 2,3,4 processes + generalization)
- SMT solvers as constraint engine
- Program Transformations
- Theorem proving (Finite model generators)

Current work

- Non atomic case → use of graphs
- Graph-based tools for network protocols (routing, broadcast)
- We are studying the properties of models: for which operations/classes of graphs we can solve problems like coverability

:

Some other applications

- Parameterized verification for biological systems
  - Bioambients and P-systems (tree structures and petri nets)
  - Conformon P-systems (petri nets + energy)
  - Kappa calculus (graph-based rules)

: