Efficient Recognition of Finite Satisfiability in UML Class Diagrams: Strengthening by Propagation of Disjoint Constraints

Azzam Maraee and Mira Balaban

Presenter: Azzam Maraee

Agenda

- Background
- Modeling Problems
- Recognition of correctness problems
  - Finite satisfiability
- Disjoint Propagation Algorithm
- Future Work

Unified Modeling Language (UML)

- OMG standard for software development.
- Consists of several languages.
- Class diagrams – most important, mostly used.
- Class diagrams – best understood:
  - Intuitively
  - Formally

Class Diagram Modeling Problems

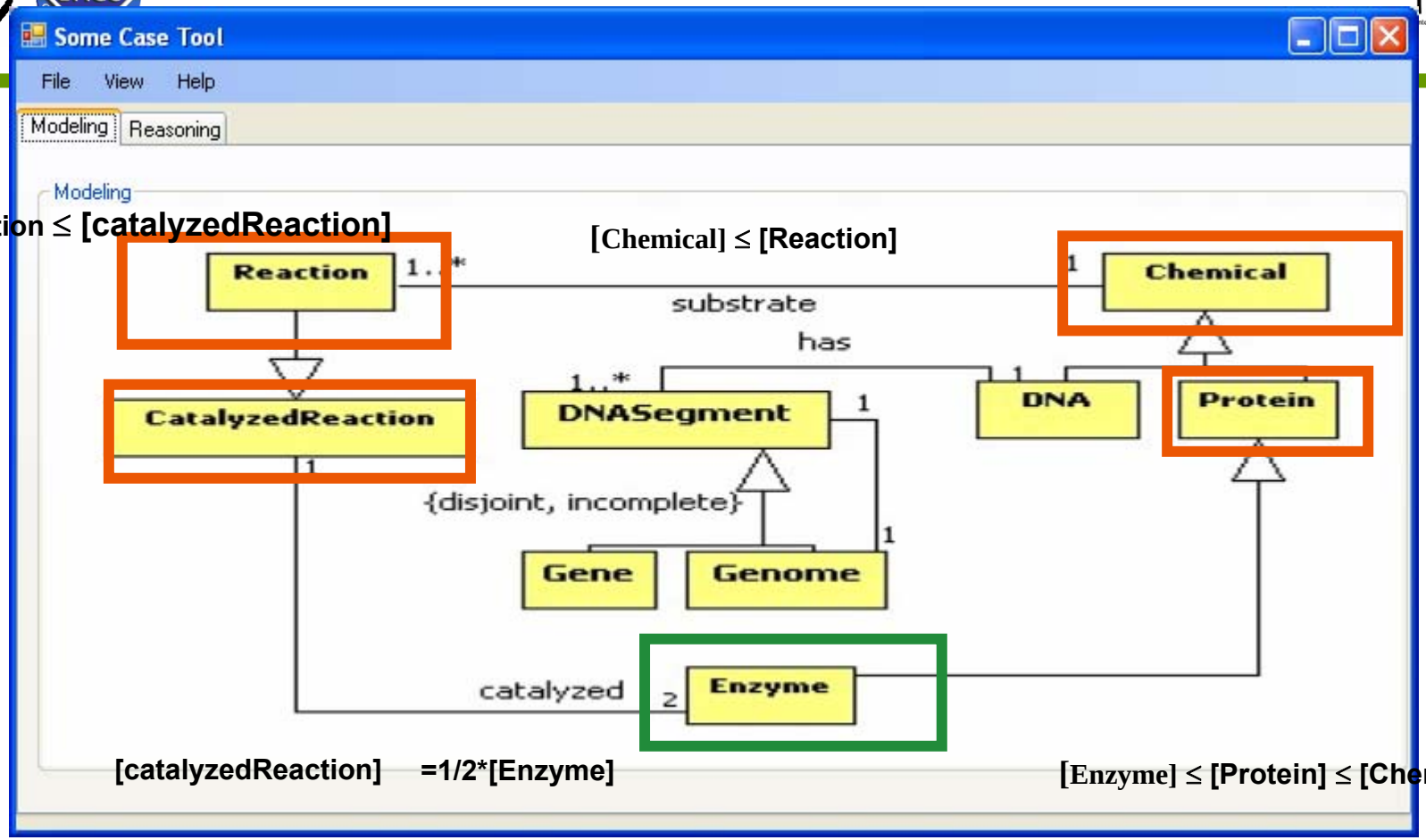
Reasons:

- Inexperienced designers create erroneous models
- Anticipate change implications
- Model defects remain undetected
- Empowered when multiple sources
  - Overlapping
  - inconsistencies

UML CASE Tools – Current Status

- Enable construction of erroneous models
- Problematic for Model Driven Engineering approach
  - No early detection of design problems
  - No identification of error causes
  - No suggestions for possible solutions
  - No advice for design improvements
- No provision of support at the level of current Integrated Development Environments (IDEs)

Example



[Reaction ≤ [catalyzedReaction]

[Chemical] ≤ [Reaction]

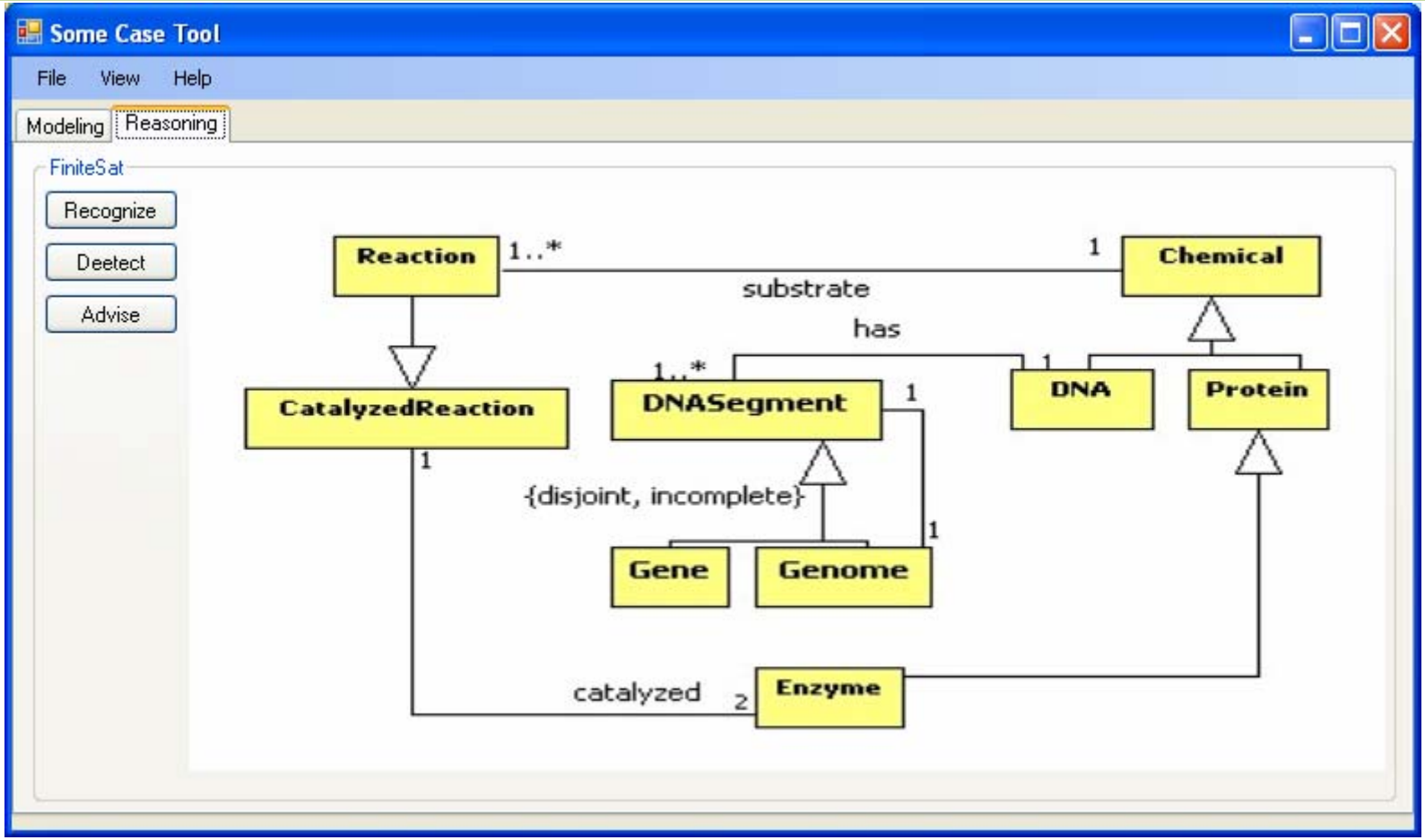
[catalyzedReaction] =1/2*[Enzyme]

[Enzyme] ≤ [Protein] ≤ [Chemical]

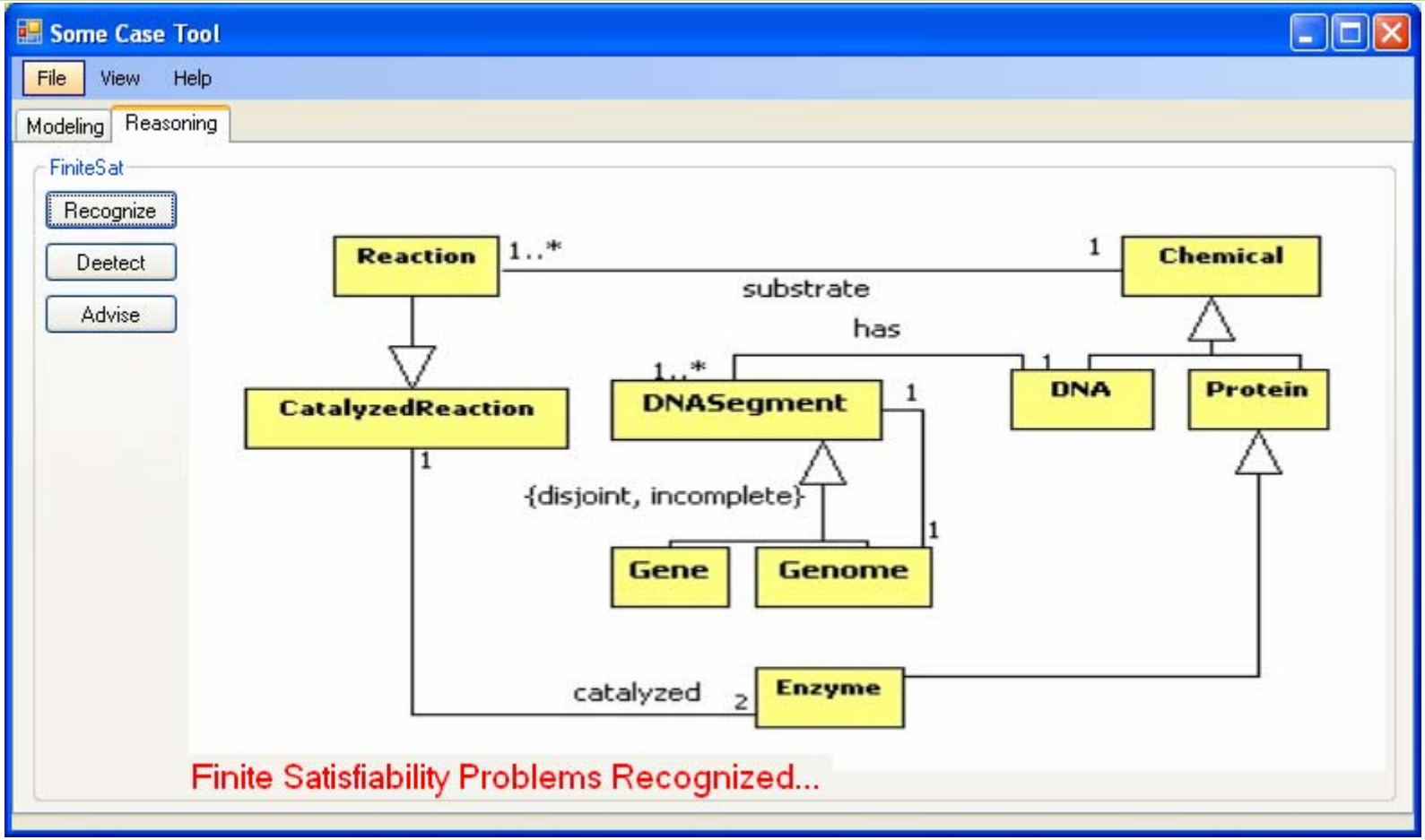
[Chemical] ≤ [Reaction] ≤ [catalyzedReaction] =1/2*[Enzyme] ≤ ½*[Protein] ≤ ½* [Chemical]

2*[Chemical]≤ [Chemical]

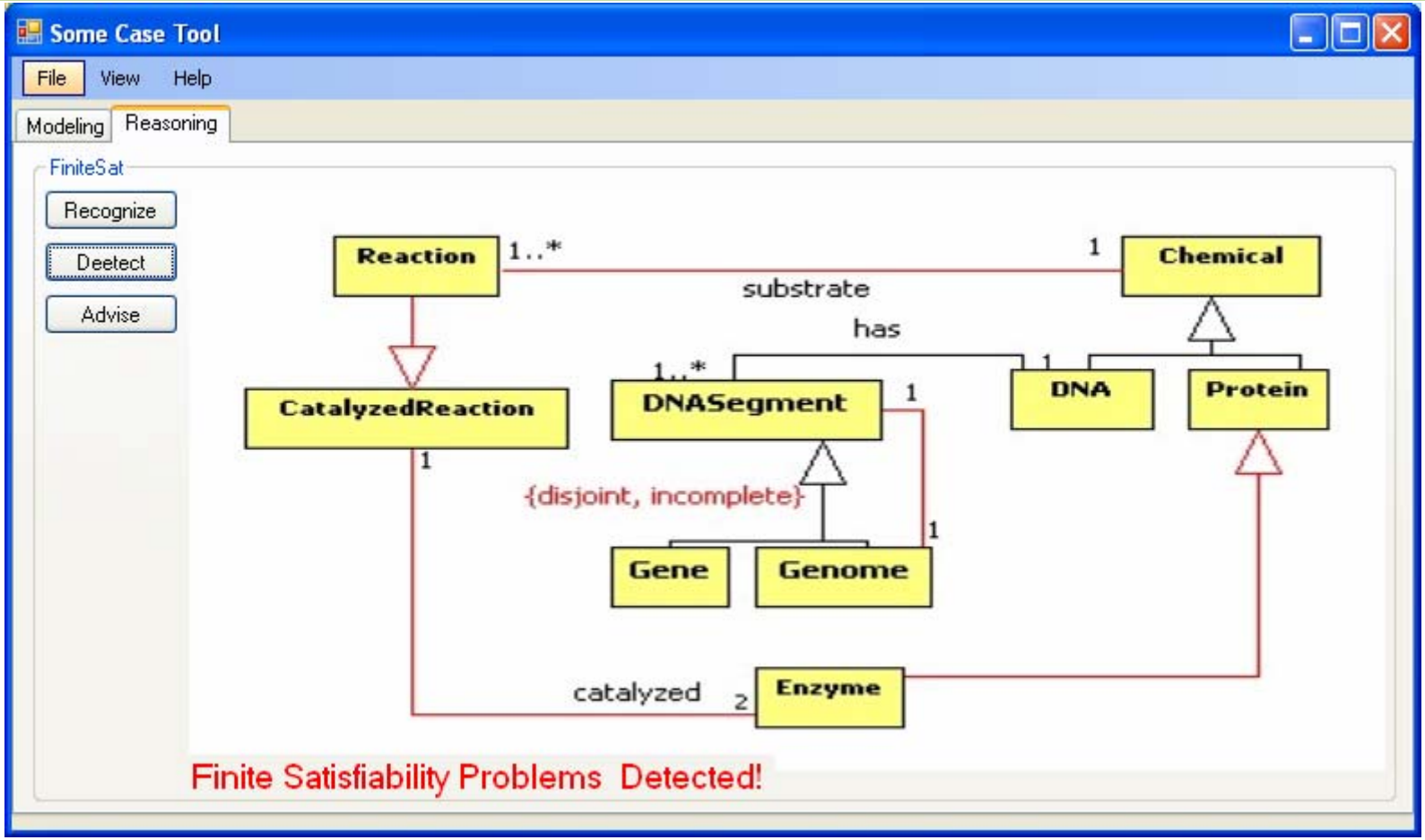
Desired Performance



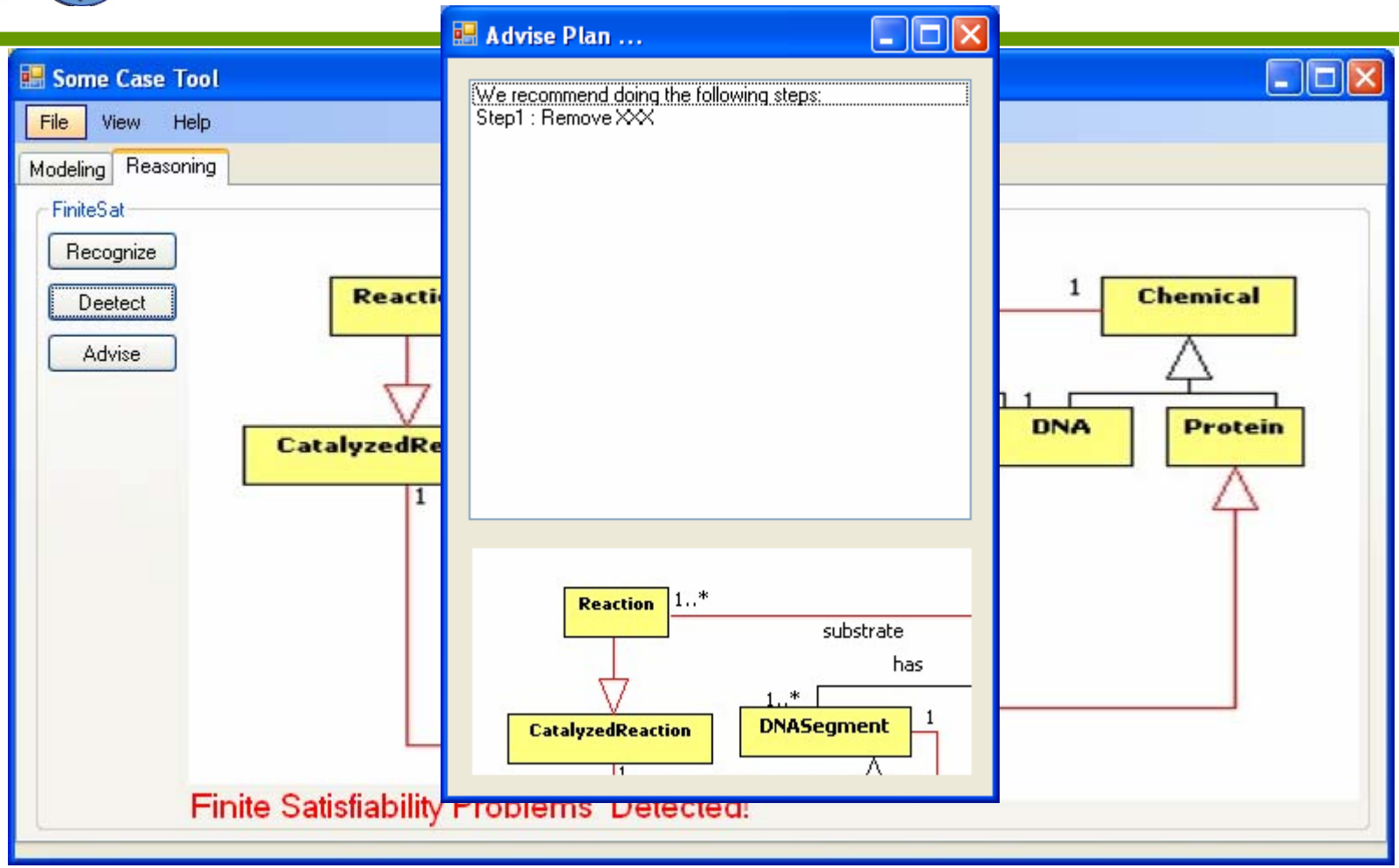
Desired Performance



Desired Performance



Desired Performance



Agenda

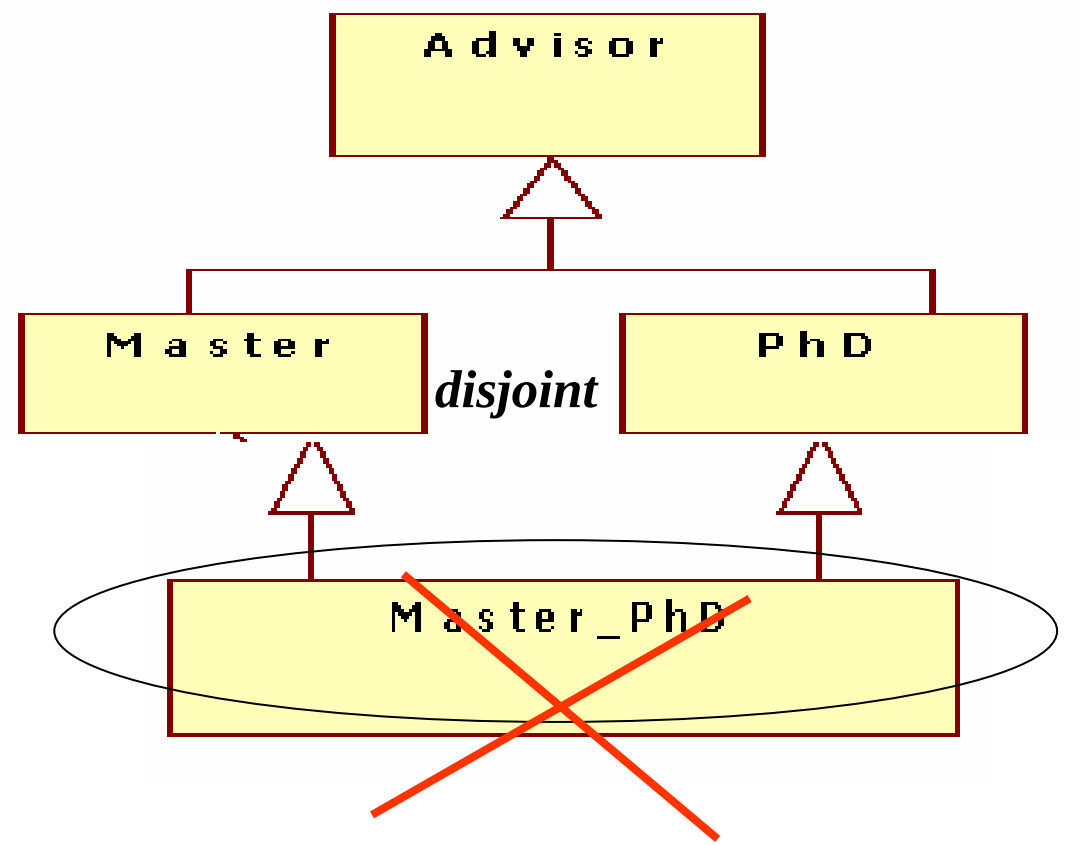
- Background
- Modeling Problems
- Recognition of correctness problems
  - Inconsistency
  - Finite satisfiability
- Disjoint Propagation Algorithm
- Conclusion and Future Work

Class Diagram Modeling Problems

- Inconsistency: Emptiness
- Finite satisfiability: Requires consistency
- Redundancy
- Incomplete design

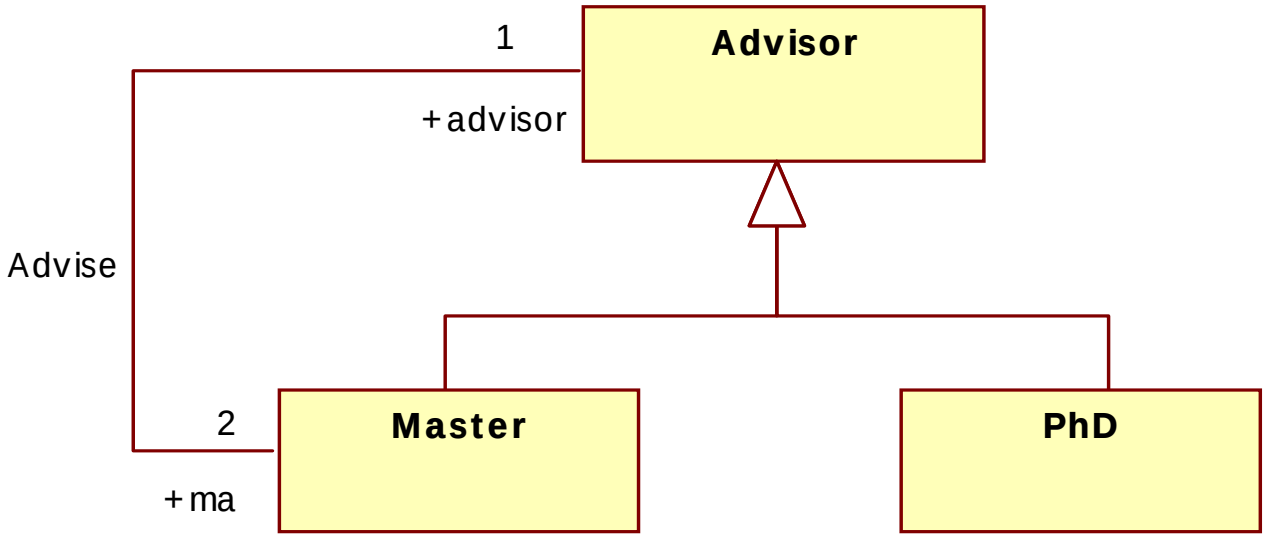
Modeling Problems – Inconsistency

- Emptiness – *Contradictory generalization set constraints*

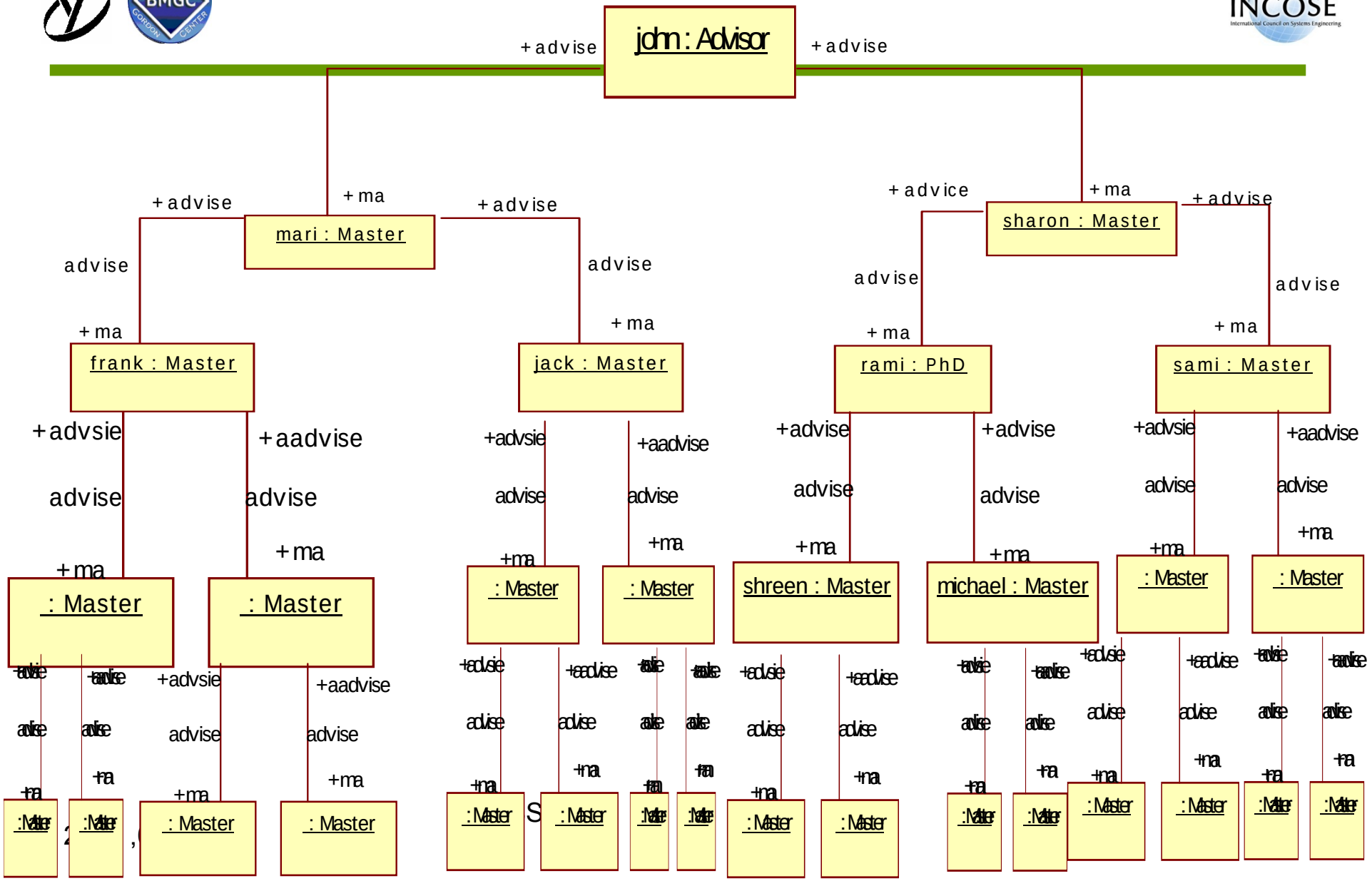


Modeling Problems – Finite Satisfiability

- Infinity – Interaction of multiplicity constraints and class hierarchy constraints: $M = A*2$, and also $M \leq A$.



Infinity Problem (2)



Semantics (1)

- A ***legal instance*** of a class diagram is an instance that satisfies all constraints.

- A ***class is consistent*** if it has a non empty extension in some legal instance.

- A ***class is finitely satisfiable*** if it has a non empty and finite extension in some finite legal instance.

Semantics (2)

- A ***class diagram is consistent*** if all of its classes are consistent.

- A ***class diagram is finitely satisfiable*** if all of its classes are finitely satisfiable.

Hardness of Finite Satisfiability

$$FSat(UML) \in EXPTIME - complete$$

Based on results in description logics:

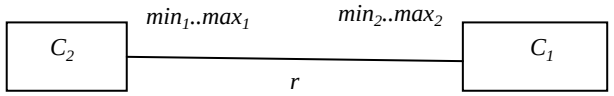
- Berardi et al. (2005)
- Lutz et al. (2005)

Agenda

- Background
- Modeling Problems
- Recognition of correctness problems
  - Finite satisfiability
- Disjoint Propagation Algorithm
- Future Work

Recognition of Finite Satisfiability methods

- ***Lenzerini and Nobily (92), Thalheim (93):***
  - ER diagrams with binary associations and Cardinality constraints.
  - Transform an ER diagram ***E*** into a system of linear inequalities ψ:



yields the inequalities:
$C1 > 0$; $C2 > 0$; $r > 0$;
For $min1 \neq 0$: $r \geq min1 \cdot C1$; for $max1 \neq \infty$: $r \leq max1 \cdot C1$;
For $min2 \neq 0$: $r \geq min2 \cdot C2$; for, $max2 \neq \infty$: $r \leq max2 \cdot C2$;
Result: *E is finitely satisfiable iff ψ is solvable. Polinomial complexity*.

Recognition of Finite Satisfiability methods

Calvanese and Lenzerini (94):

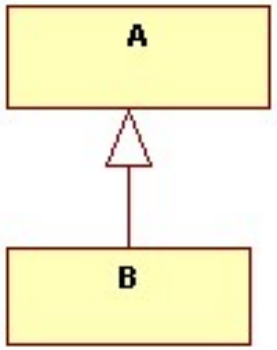
- Extension with Class Hierarchy constraints.
- Transform an ER diagram ***E*** into a different system of linear inequalities ψ.
- Result: ***E*** is finitely satisfiable iff ψ is solvable.
- Size is exponential!

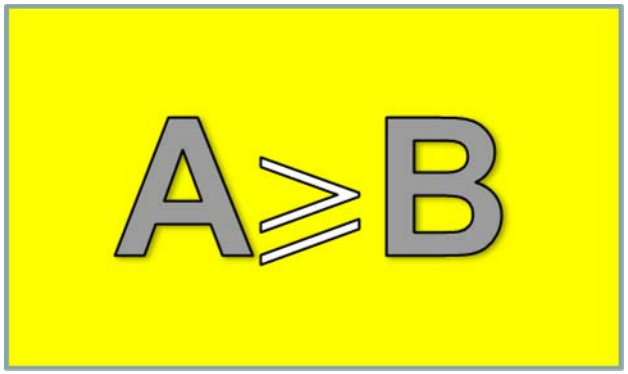
Recognition of Finite Satisfiability methods

- ***Maraee and Balaban (2007):***
  - class diagrams with:
    - binary associations,
    - Cardinality constraints,
    - Class Hierarchy,
    - Generalization Set constraints (disjoint, complete, overlap, incomplete),
    - Association class constraints,
    - Qualifier constraints

The FiniteSat Algorithm: An Example

- Class hierarchy constrain

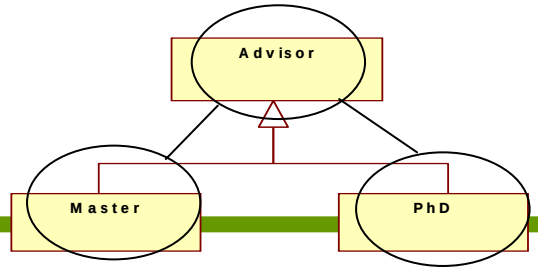




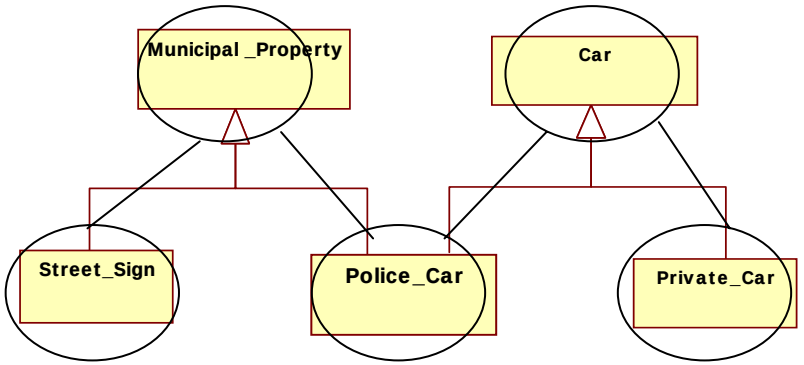
Recognition of Finite Satisfiability methods

- ***Maraee and Balaban (2007):***
  - ***FiniteSat*** algorithm: Transforms a class diagram ***C*** into a system of linear inequalities ψ.
  - Results – Based on ***class hierarchy structure:***
    - ***Tree***
    - ***Acyclic***
    - ***Graph***

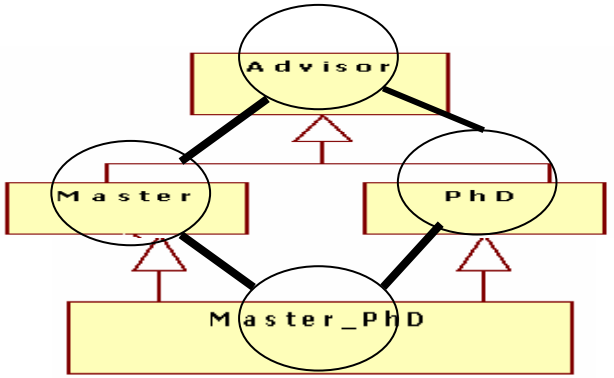
Tree:



- Acyclic:

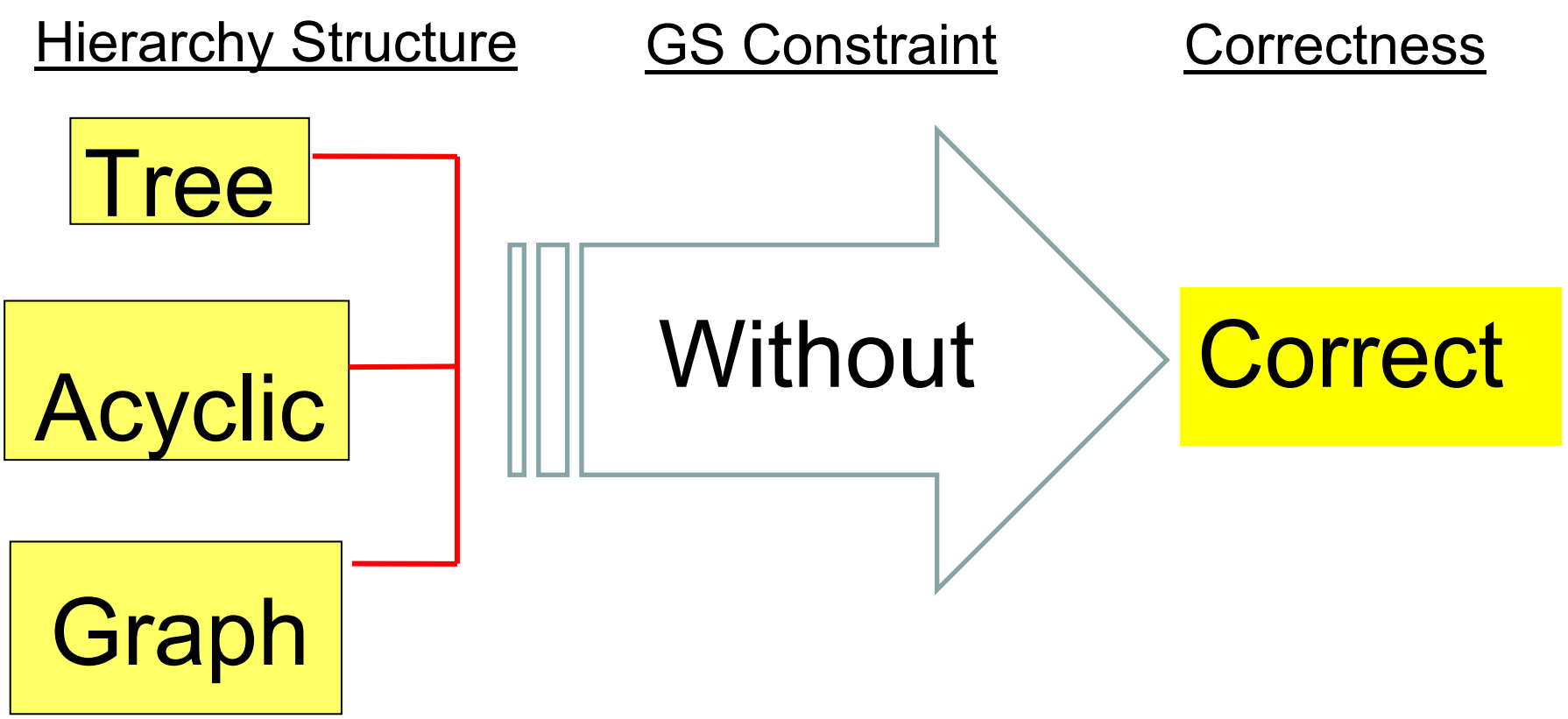


- Graph:



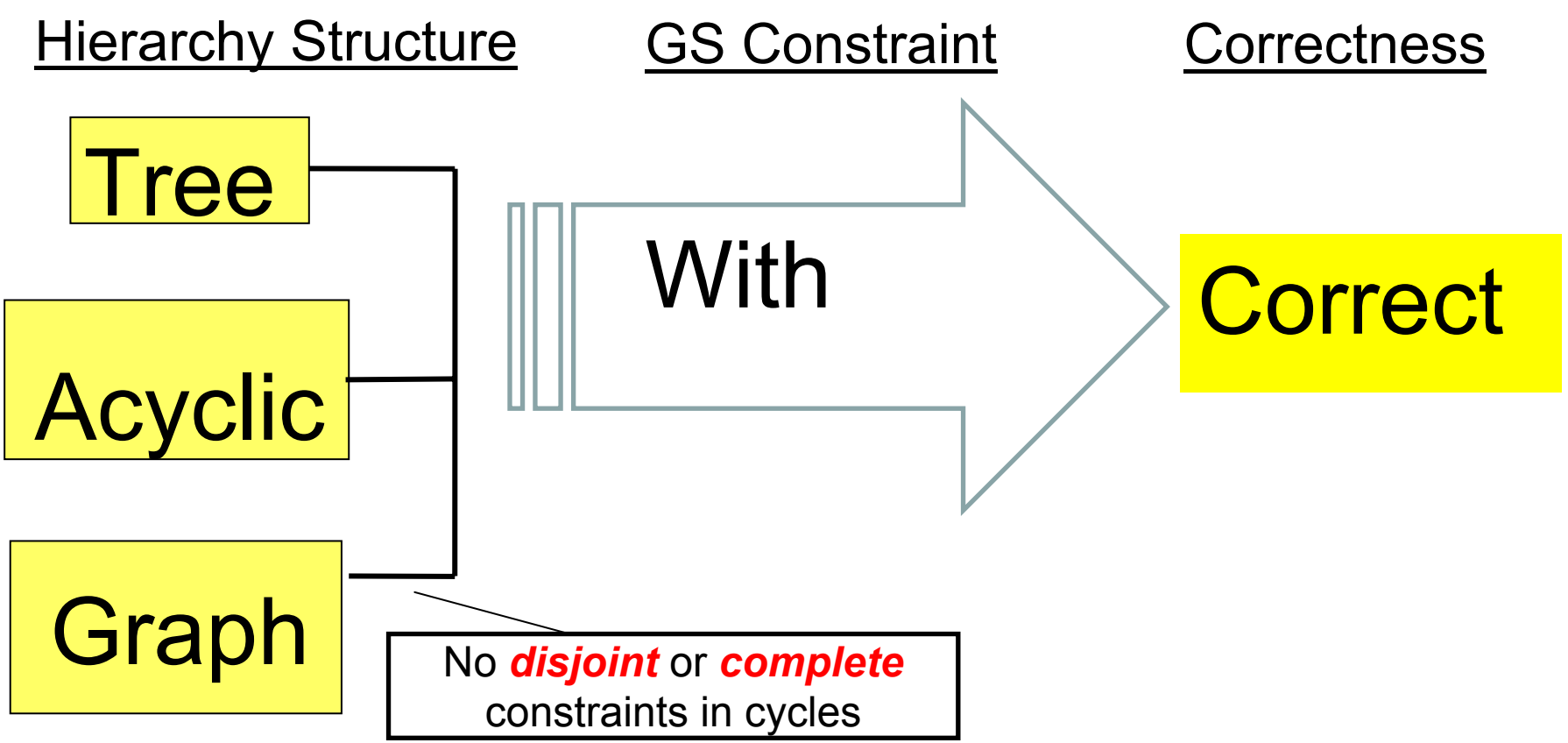
Result:

Theorem: CD is finitely satisfiabile iff ψ is solvable.



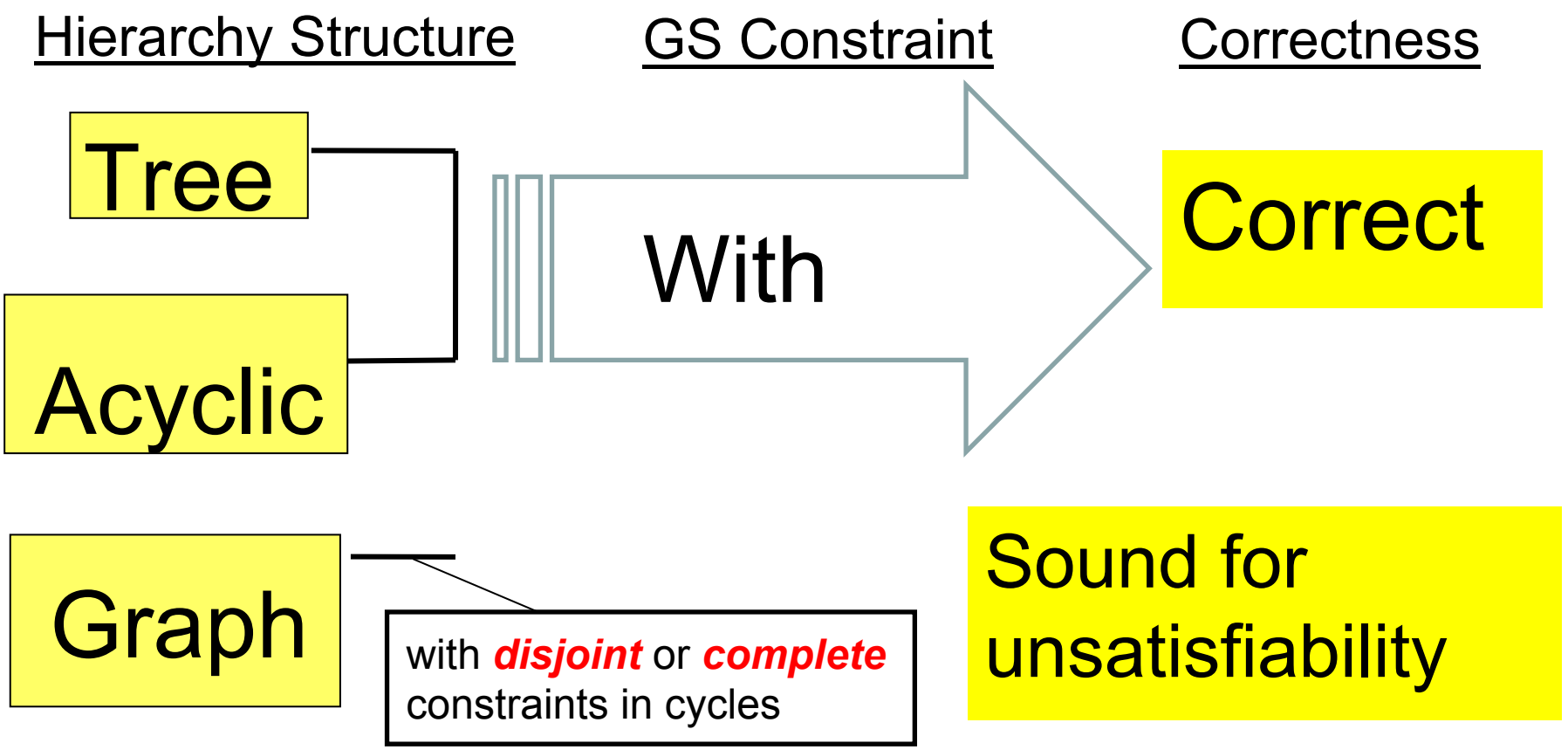
Result:

Theorem: ***CD*** *is finitely satisfiabile iff ψ is solvable.*

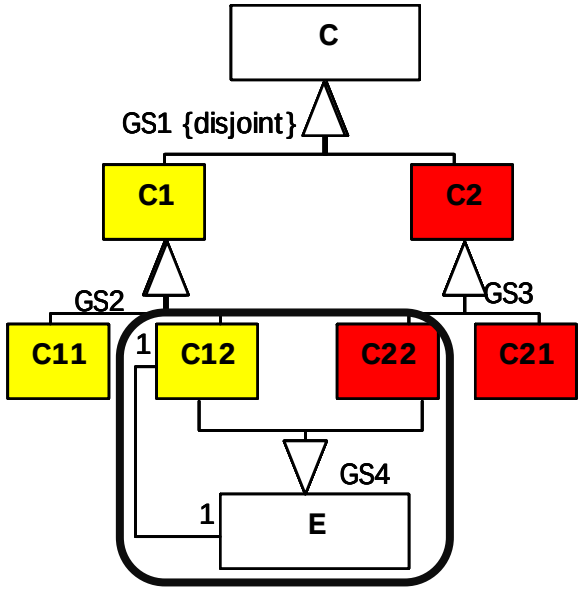


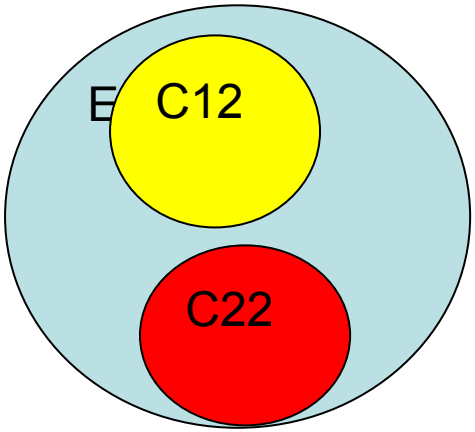
Result:

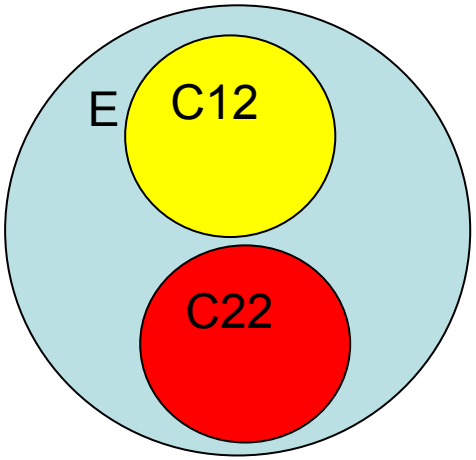
Theorem: ***CD*** *is finitely satisfiabile iff ψ is solvable*



Limitation of the FiniteSat Algorithm: An Example





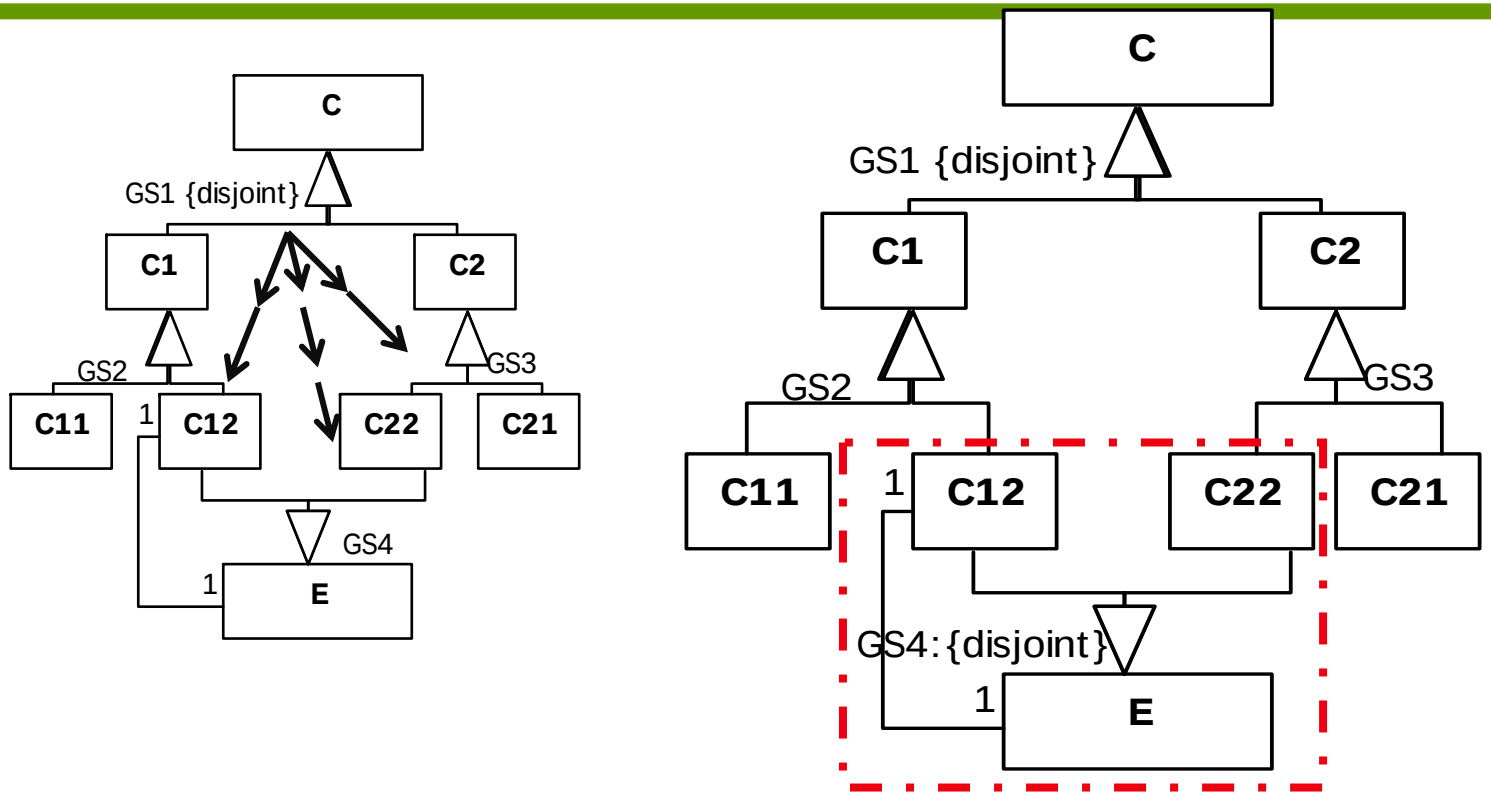


The only solution for proper set inclusion with equal size is that the sets are either empty or infinite.

Agenda

- Background
- Modeling Problems
- Recognition of correctness problems
  - Finite satisfiability
- Disjoint Propagation Algorithm
- Future Work

FiniteSat is strengthened by propagation of disjoint and incomplete GS constraints.

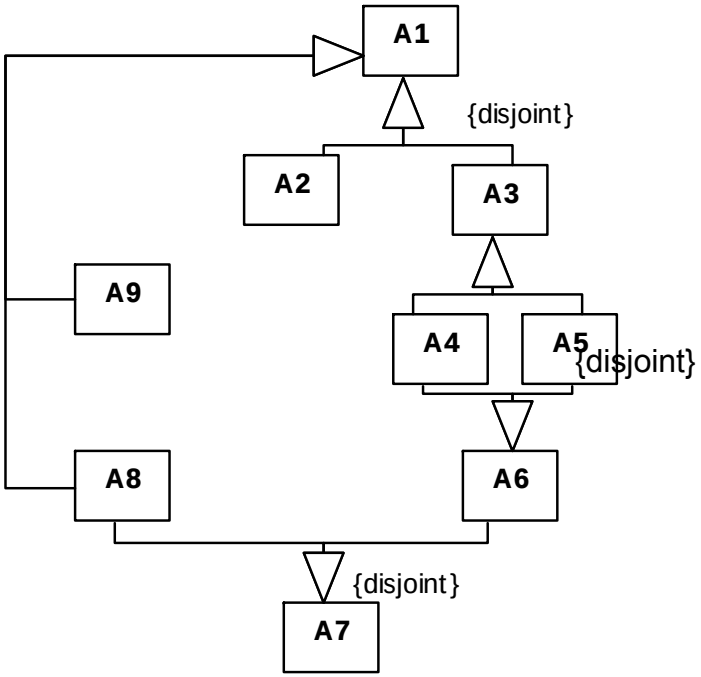


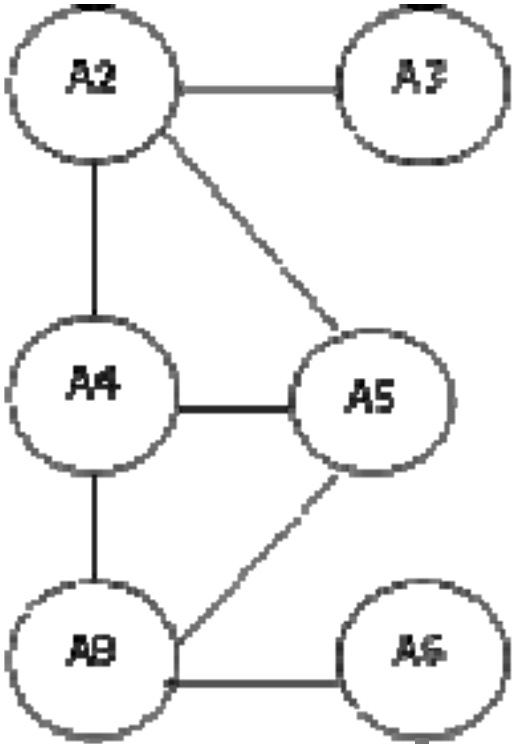
Before: ***FiniteSat*** does not detect the finite satisfiability problem

After: ***FiniteSat*** detects the finite satisfiability problem

Disjoint Propagation Algorithm: Definition

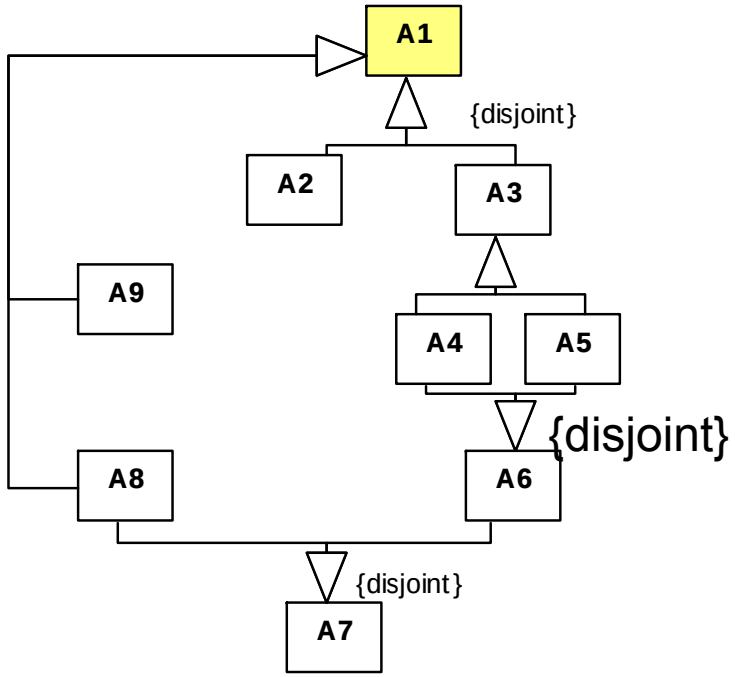
Disjoint graph: is an undirected graph whose nodes represent the classes and its edges connect nodes of disjoint classes

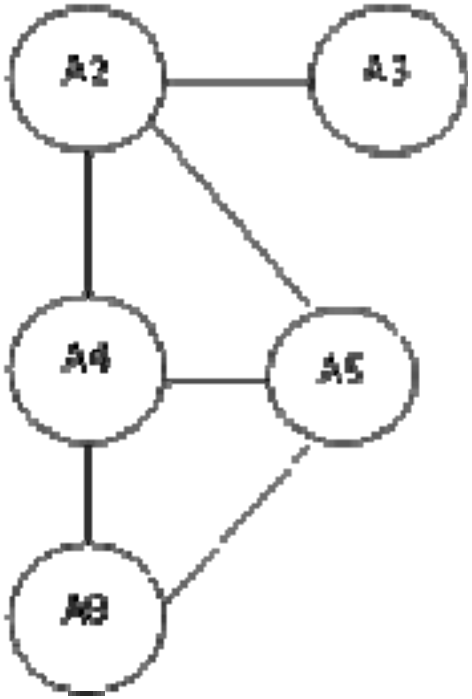




Disjoint Propagation Algorithm: An Example

1. Create the disjoint graph for each class*

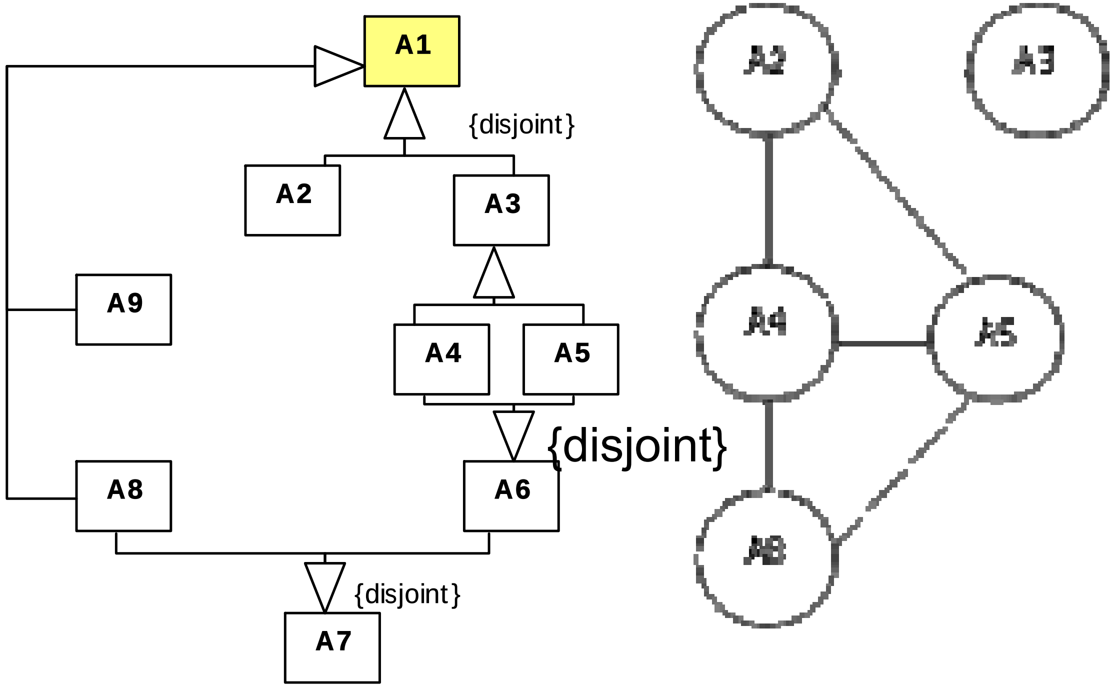


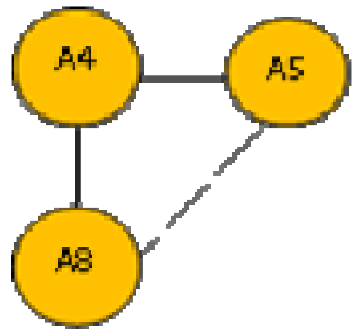


**** The restriction of the disjoint graph to the set of proper descendents the class***

Disjoint Propagation Algorithm: An Example

2. Select the maximal cliques



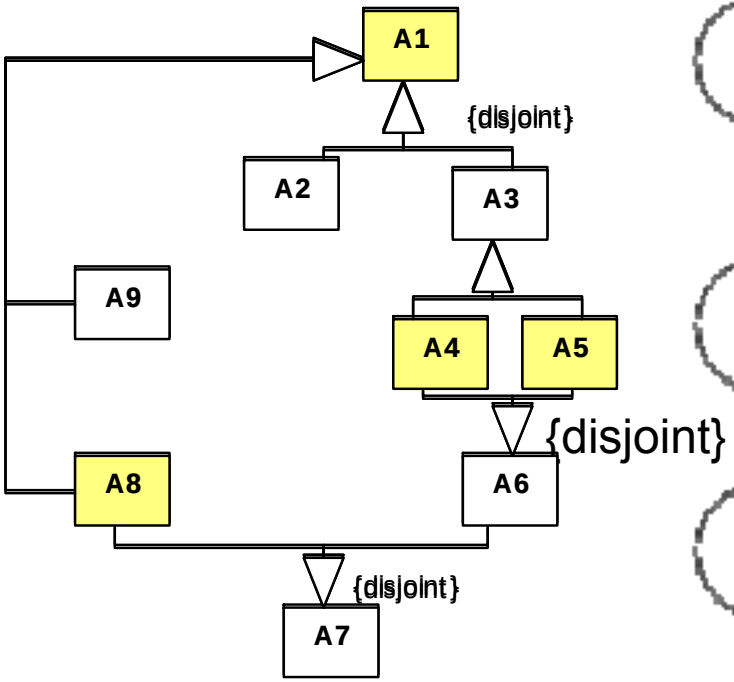


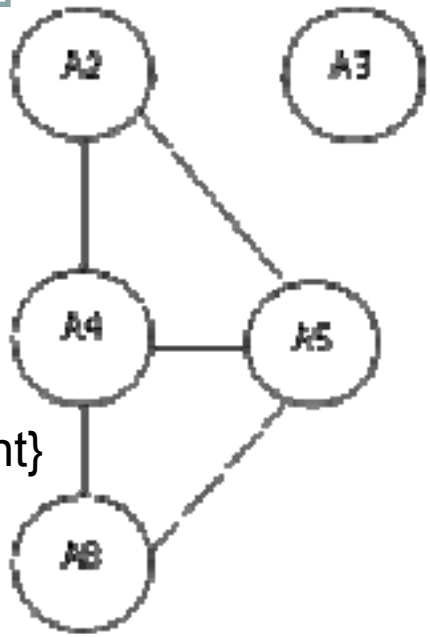
3. Order the maximal cliques according to class hierarchy relation

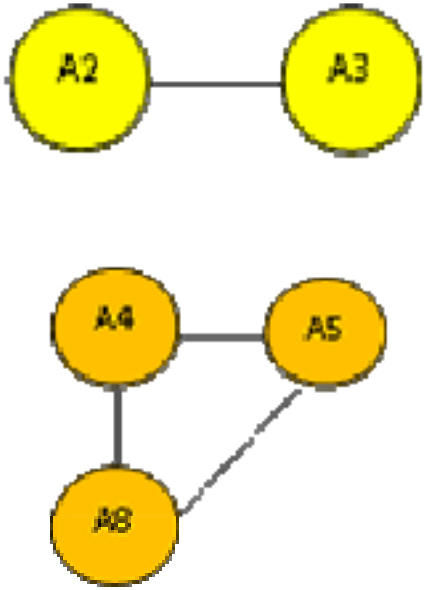
4. Select the top element in the ordered set of maximal cliques

Disjoint Propagation Algorithm: An Example

Gs(***A***1, ***A***4, ***A***5, ***A***8; ***disjoint***}



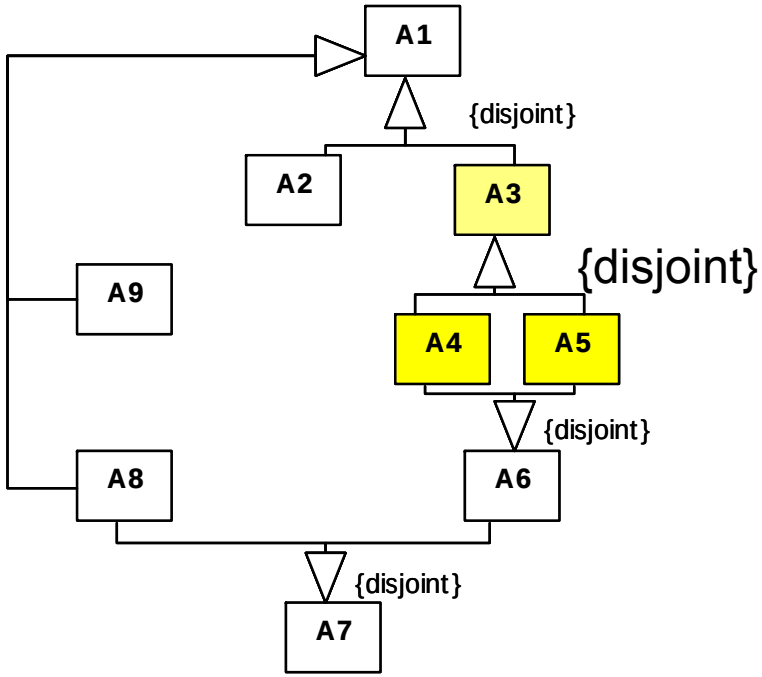


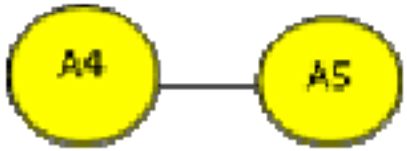


4. Extract the top sub-diagram with disjoint GS of maximal disjoint descendents, for all classes

5. Execute the above algorithm on all selected sets of disjoint

Disjoint Propagation Algorithm: An Example





Gs(***A3***, ***A***4, ***A***5; ***disjoint***}

Future Work

- Handle class diagram versioning: Incremental ***FiniteSat***
- Refinement: FiniteSat with class hierarchy cycles with *disjoint* or *complete* constraints.
- Extend FiniteSat to full UML CDs: N-ary associations, aggregation, association constraints.
- Implementation in a UML case tool:
  - Establish a platform for extensible implementation.
  - Show scalability of FiniteSat and the detection methods.
    - Implementation is ongoing.
    - Apply to large known examples.
- Combine the detection methods with Warning and Advice
  - Heuristics based on patterns.
  - Consult domain ontologies.
- Combine with correctness methods for other modeling artifacts