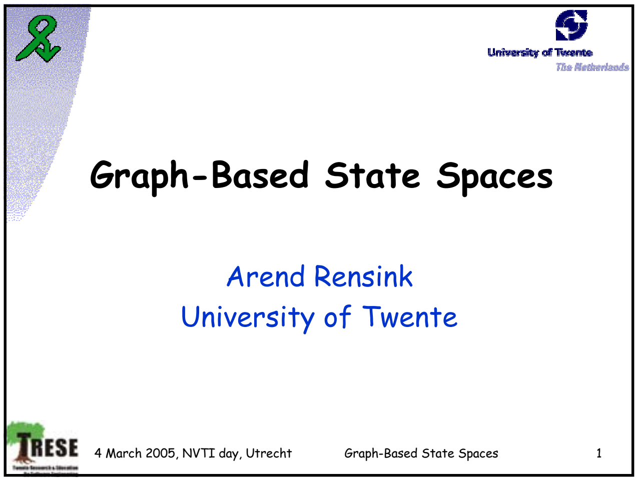
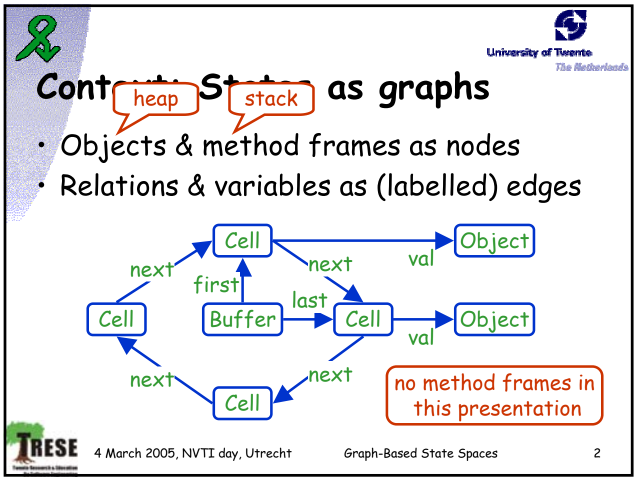


University of Twente
The Netherlands

Graph-Based State Spaces

Arend Rensink
University of Twente

TRESE



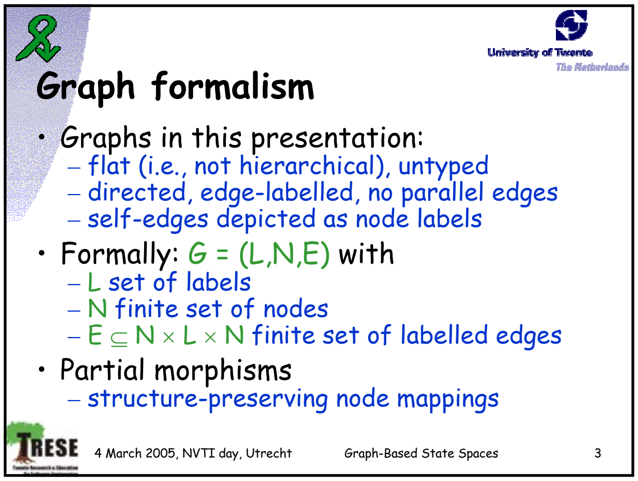
University of Twente
The Netherlands

Content: States as graphs

- Objects & method frames as nodes
- Relations & variables as (labelled) edges

no method frames in this presentation

TRESE

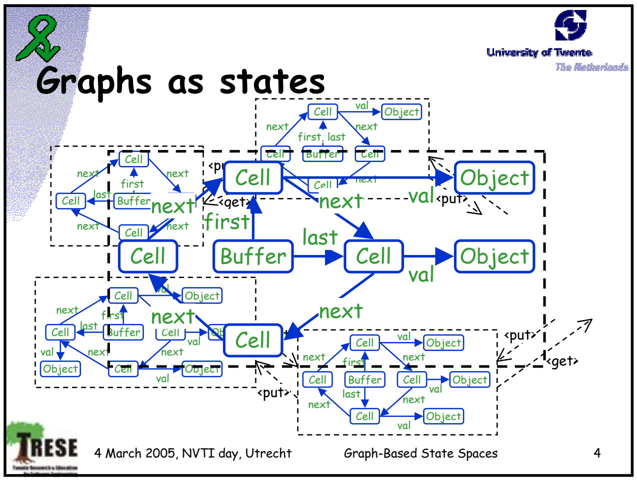


University of Twente
The Netherlands

Graph formalism

- Graphs in this presentation:
  - flat (i.e., not hierarchical), untyped
  - directed, edge-labelled, no parallel edges
  - self-edges depicted as node labels
- Formally: $G = (L,N,E)$ with
  - L set of labels
  - N finite set of nodes
  - $E \subseteq N \times L \times N$ finite set of labelled edges
- Partial morphisms
  - structure-preserving node mappings

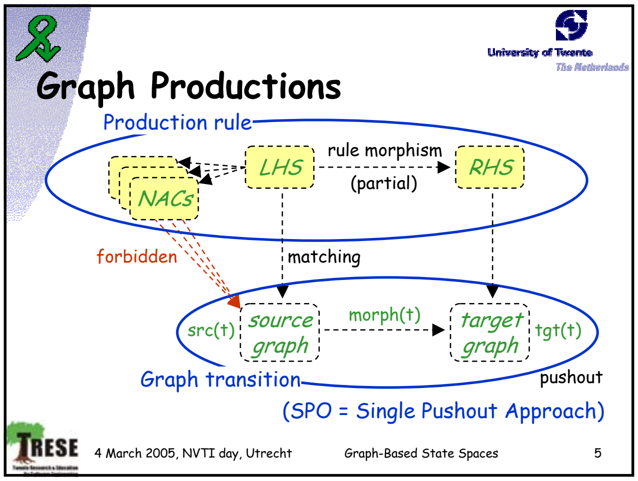
TRESE



University of Twente
The Netherlands

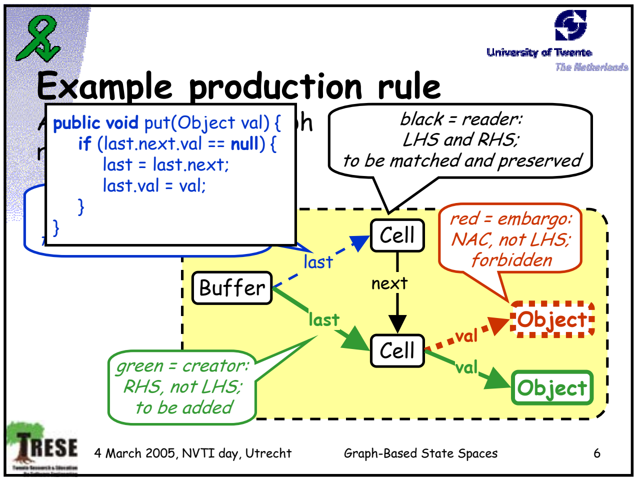
Graphs as states

TRESE



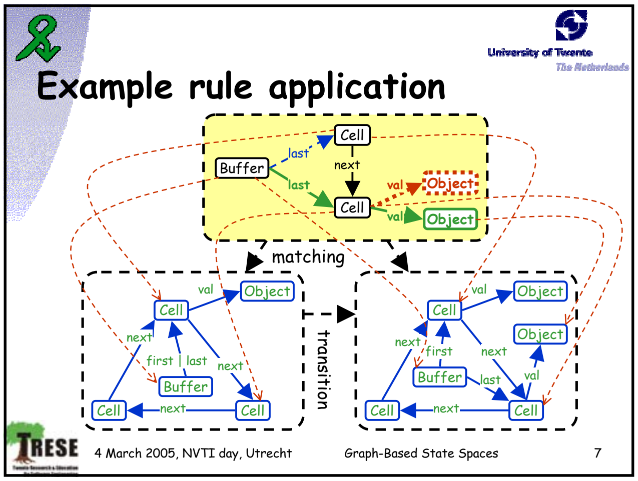
Graph Productions

(SPO = Single Pushout Approach)

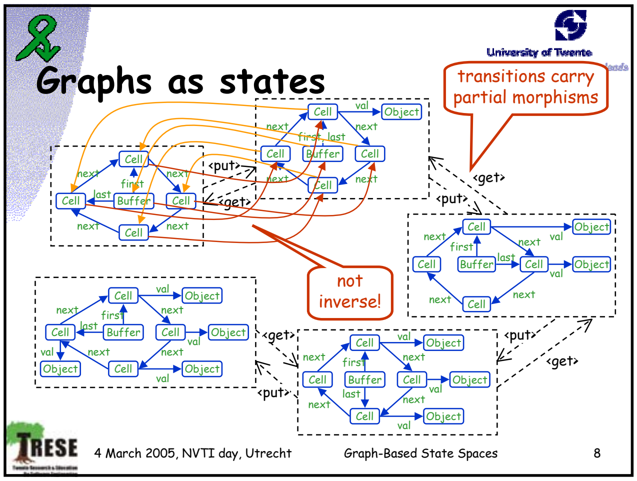


Example production rule

```
public void put(Object val) {
    if (last.next.val == null) {
        last = last.next;
        last.val = val;
    }
}
```



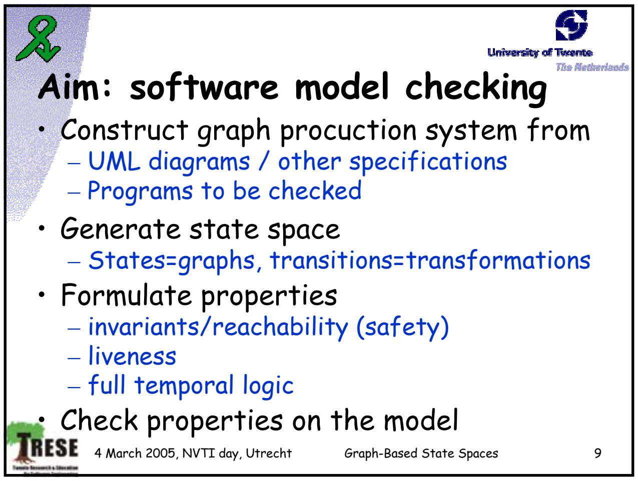
Example rule application



Graphs as states

transitions carry partial morphisms

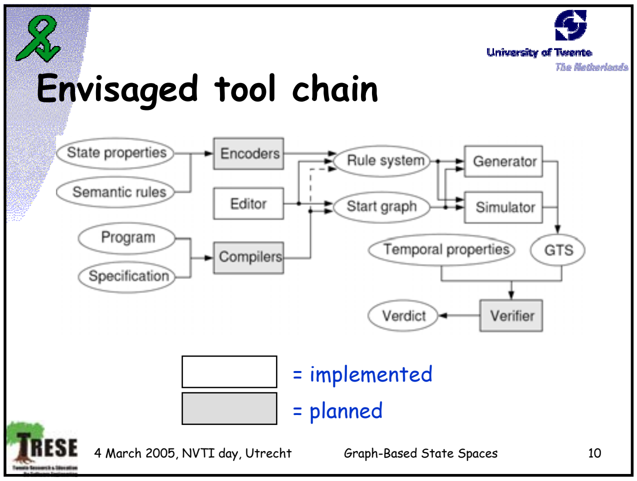
not inverse!



University of Twente
The Netherlands

Aim: software model checking

- Construct graph procuction system from
  - UML diagrams / other specifications
  - Programs to be checked
- Generate state space
  - States=graphs, transitions=transformations
- Formulate properties
  - invariants/reachability (safety)
  - liveness
  - full temporal logic
- Check properties on the model

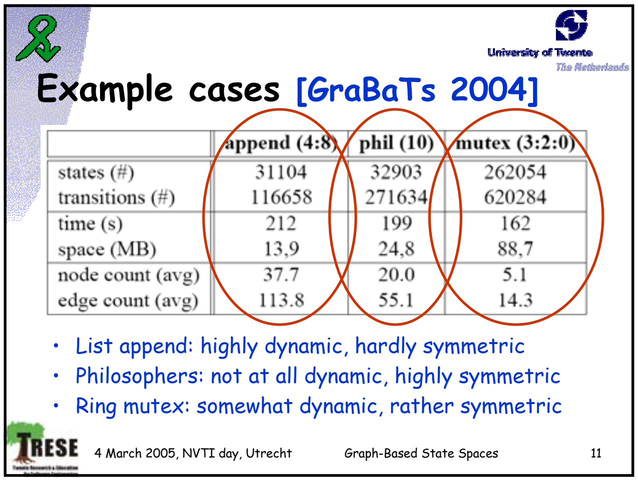


University of Twente
The Netherlands

Envisaged tool chain

□ = implemented

▨ = planned

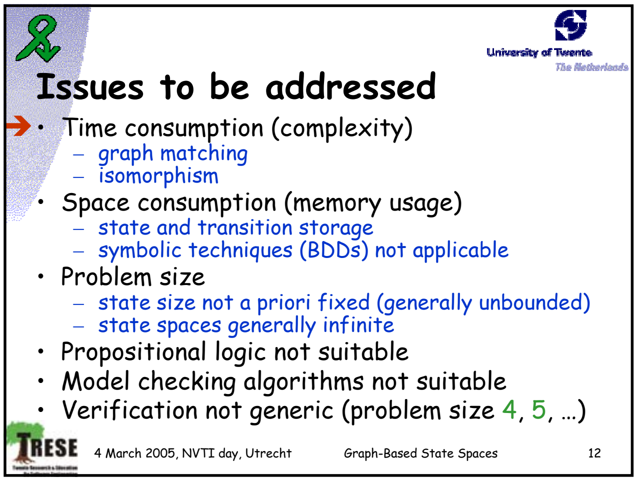


University of Twente
The Netherlands

Example cases [GraBaTs 2004]

	append (4:8)	phil (10)	mutex (3:2:0)
states (#)	31104	32903	262054
transitions (#)	116658	271634	620284
time (s)	212	199	162
space (MB)	13,9	24,8	88,7
node count (avg)	37.7	20.0	5.1
edge count (avg)	113.8	55.1	14.3

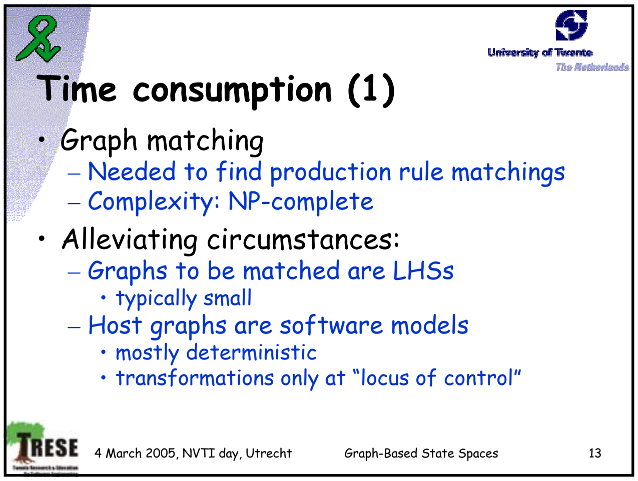
- List append: highly dynamic, hardly symmetric
- Philosophers: not at all dynamic, highly symmetric
- Ring mutex: somewhat dynamic, rather symmetric



University of Twente
The Netherlands

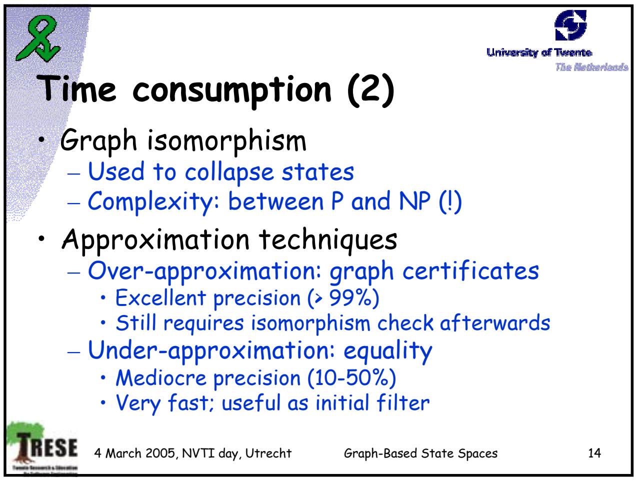
Issues to be addressed

- Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs) not applicable
- Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- Propositional logic not suitable
- Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)



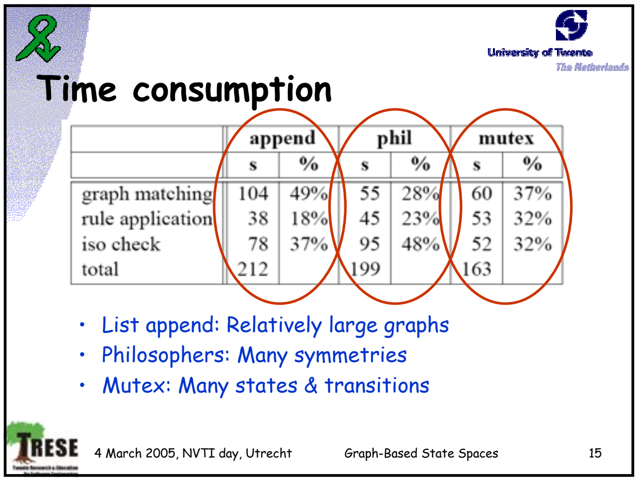
Time consumption (1)

- Graph matching
  - Needed to find production rule matchings
  - Complexity: NP-complete
- Alleviating circumstances:
  - Graphs to be matched are LHSs
    - typically small
  - Host graphs are software models
    - mostly deterministic
    - transformations only at "locus of control"



Time consumption (2)

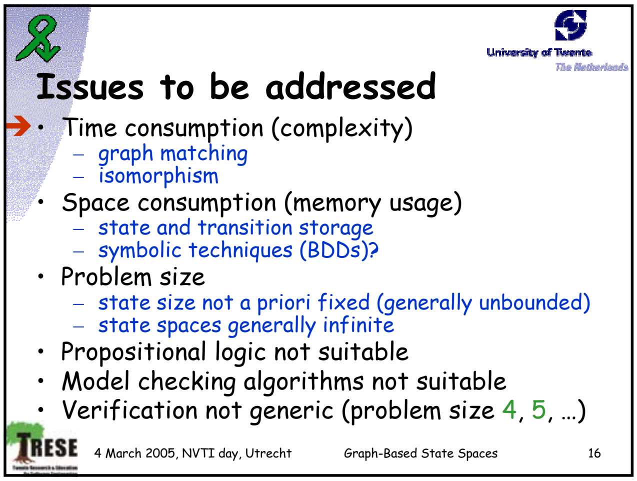
- Graph isomorphism
  - Used to collapse states
  - Complexity: between P and NP (!)
- Approximation techniques
  - Over-approximation: graph certificates
    - Excellent precision (> 99%)
    - Still requires isomorphism check afterwards
  - Under-approximation: equality
    - Mediocre precision (10-50%)
    - Very fast; useful as initial filter



Time consumption

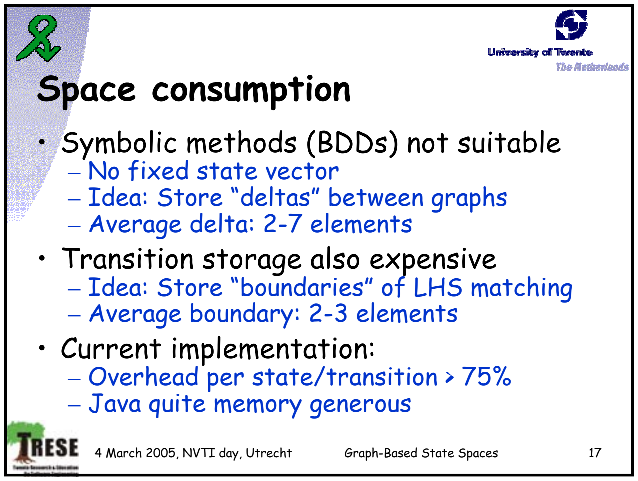
	append		phil		mutex	
	s	%	s	%	s	%
graph matching	104	49%	55	28%	60	37%
rule application	38	18%	45	23%	53	32%
iso check	78	37%	95	48%	52	32%
total	212		199		163	

- List append: Relatively large graphs
- Philosophers: Many symmetries
- Mutex: Many states & transitions



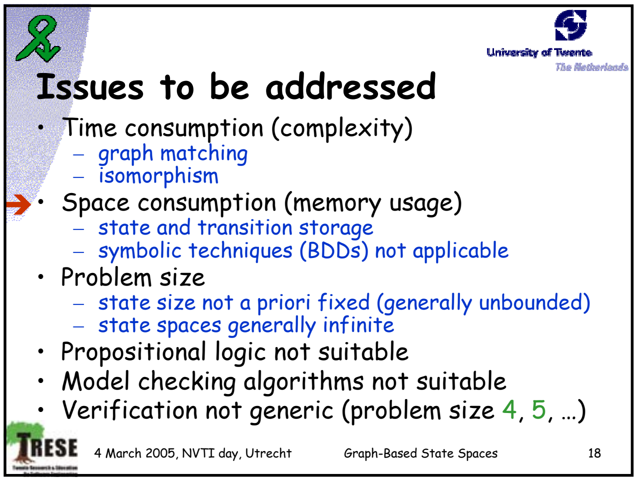
Issues to be addressed

- → Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs)?
- Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- Propositional logic not suitable
- Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)



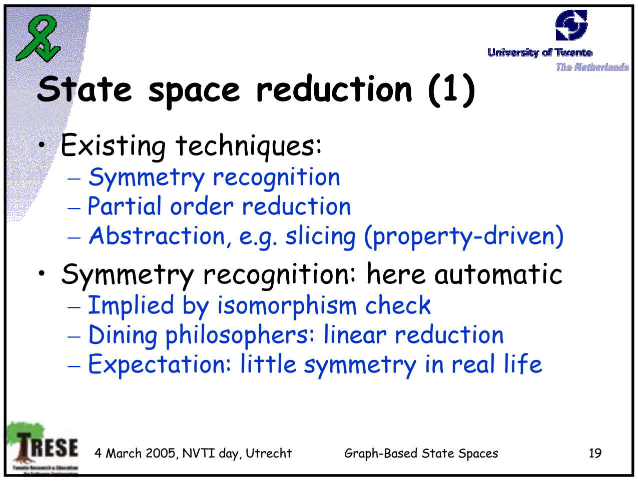
Space consumption

- Symbolic methods (BDDs) not suitable
  - No fixed state vector
  - Idea: Store "deltas" between graphs
  - Average delta: 2-7 elements
- Transition storage also expensive
  - Idea: Store "boundaries" of LHS matching
  - Average boundary: 2-3 elements
- Current implementation:
  - Overhead per state/transition > 75%
  - Java quite memory generous



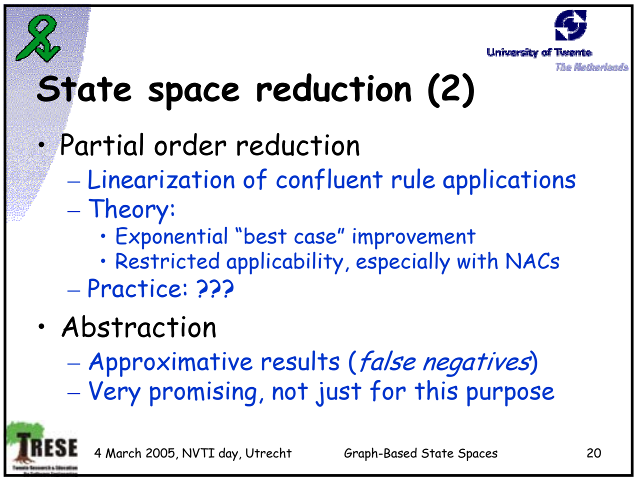
Issues to be addressed

- Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs) not applicable
- Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- Propositional logic not suitable
- Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)



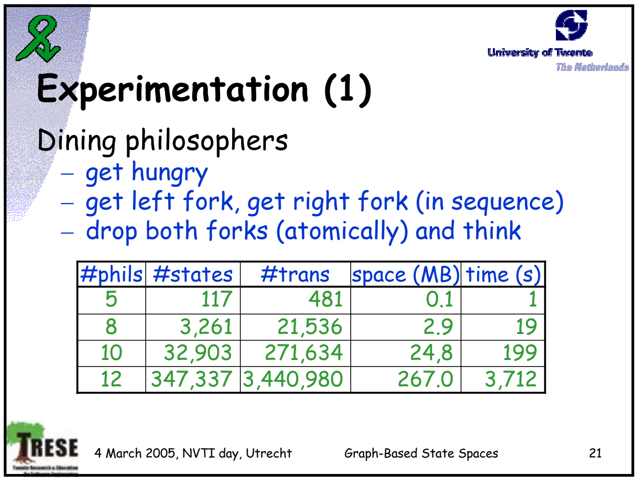
State space reduction (1)

- Existing techniques:
  - Symmetry recognition
  - Partial order reduction
  - Abstraction, e.g. slicing (property-driven)
- Symmetry recognition: here automatic
  - Implied by isomorphism check
  - Dining philosophers: linear reduction
  - Expectation: little symmetry in real life



State space reduction (2)

- Partial order reduction
  - Linearization of confluent rule applications
  - Theory:
    - Exponential "best case" improvement
    - Restricted applicability, especially with NACs
  - Practice: ???
- Abstraction
  - Approximative results (*false negatives*)
  - Very promising, not just for this purpose

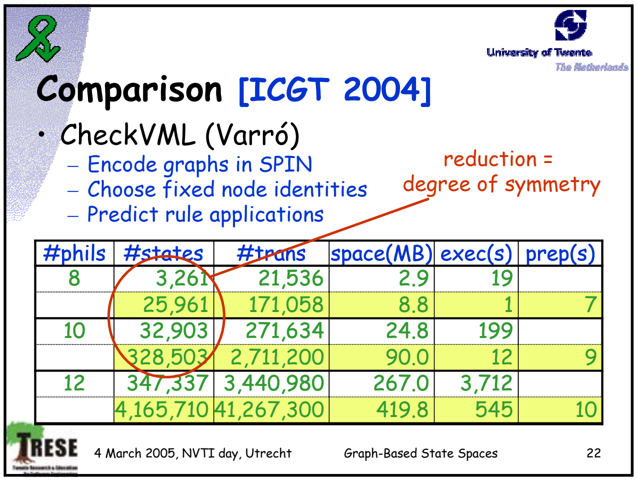


Experimentation (1)

Dining philosophers

- get hungry
- get left fork, get right fork (in sequence)
- drop both forks (atomically) and think

#phils	#states	#trans	space (MB)	time (s)
5	117	481	0.1	1
8	3,261	21,536	2.9	19
10	32,903	271,634	24,8	199
12	347,337	3,440,980	267.0	3,712

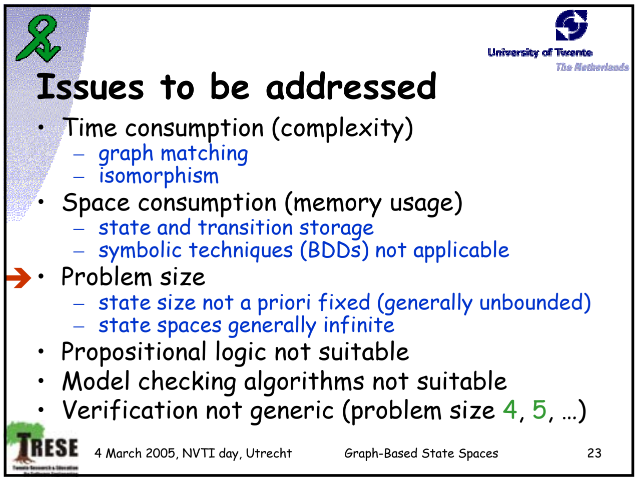


Comparison [ICGT 2004]

- CheckVML (Varró)
  - Encode graphs in SPIN
  - Choose fixed node identities
  - Predict rule applications

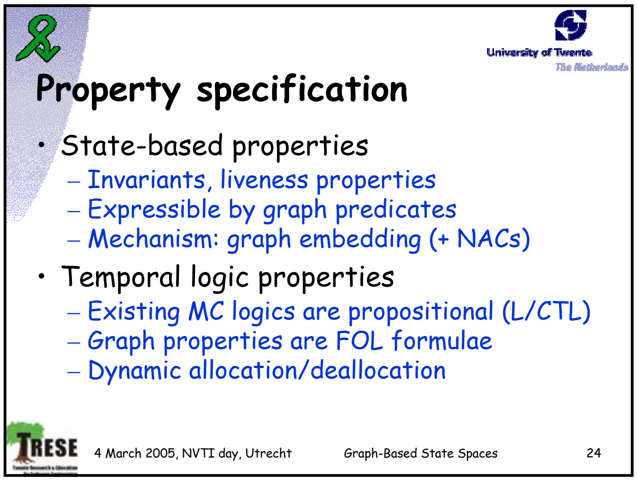
reduction =
degree of symmetry

#phils	#states	#trans	space(MB)	exec(s)	prep(s)
8	3,261	21,536	2.9	19	
	25,961	171,058	8.8	1	7
10	32,903	271,634	24.8	199	
	328,503	2,711,200	90.0	12	9
12	347,337	3,440,980	267.0	3,712	
	4,165,710	41,267,300	419.8	545	10



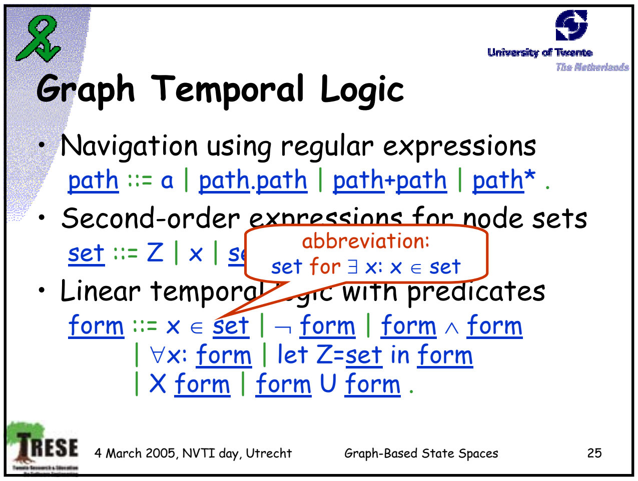
Issues to be addressed

- Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs) not applicable
- ➔ Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- Propositional logic not suitable
- Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)



Property specification

- State-based properties
  - Invariants, liveness properties
  - Expressible by graph predicates
  - Mechanism: graph embedding (+ NACs)
- Temporal logic properties
  - Existing MC logics are propositional (L/CTL)
  - Graph properties are FOL formulae
  - Dynamic allocation/deallocation



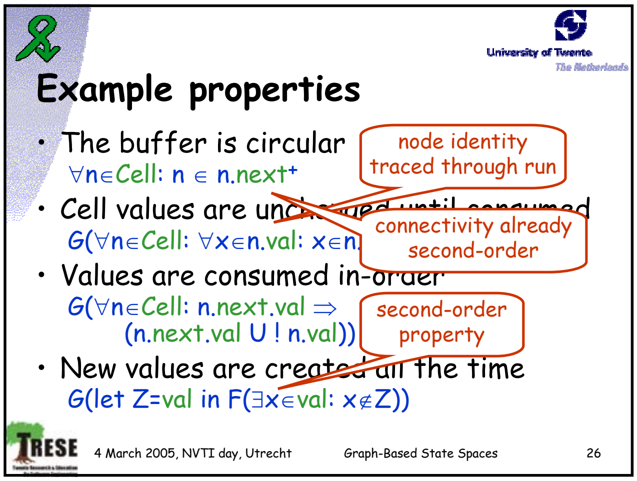
University of Twente
The Netherlands

Graph Temporal Logic

- Navigation using regular expressions
  $\underline{path} ::= a \mid \underline{path}.\underline{path} \mid \underline{path}+\underline{path} \mid \underline{path}^*$.
- Second-order expressions for node sets
  $\underline{set} ::= Z \mid x \mid \underline{se}$
- Linear temporal logic with predicates
  $\underline{form} ::= x \in \underline{set} \mid \neg\ \underline{form} \mid \underline{form} \wedge \underline{form}$
  $\mid \forall x{:}\ \underline{form} \mid \text{let } Z{=}\underline{set} \text{ in } \underline{form}$
  $\mid X\ \underline{form} \mid \underline{form}\ U\ \underline{form}$.

abbreviation:
set for $\exists\, x{:}\ x \in$ set

TRESE 4 March 2005, NVTI day, Utrecht Graph-Based State Spaces 25



University of Twente
The Netherlands

Example properties

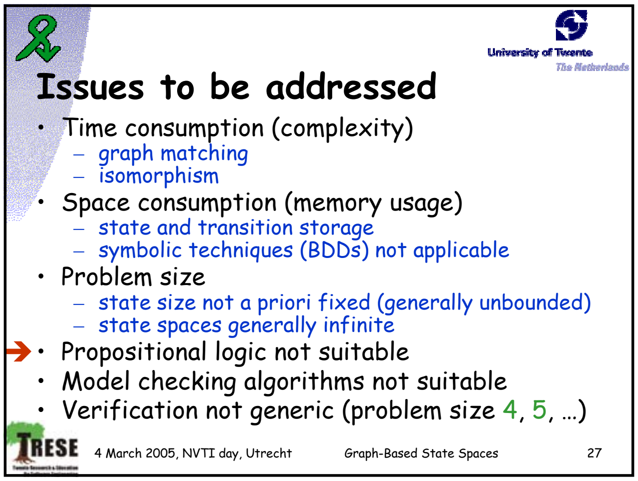
- The buffer is circular
  $\forall n{\in}Cell{:}\ n \in n.next^{+}$
- Cell values are unchanged until consumed
  $G(\forall n{\in}Cell{:}\ \forall x{\in}n.val{:}\ x{\in}n.$
- Values are consumed in-order
  $G(\forall n{\in}Cell{:}\ n.next.val \Rightarrow$
  $(n.next.val\ U\ !\ n.val))$
- New values are created all the time
  $G(\text{let } Z{=}val \text{ in } F(\exists x{\in}val{:}\ x{\notin}Z))$

node identity
traced through run

connectivity already
second-order

second-order
property

TRESE 4 March 2005, NVTI day, Utrecht Graph-Based State Spaces 26

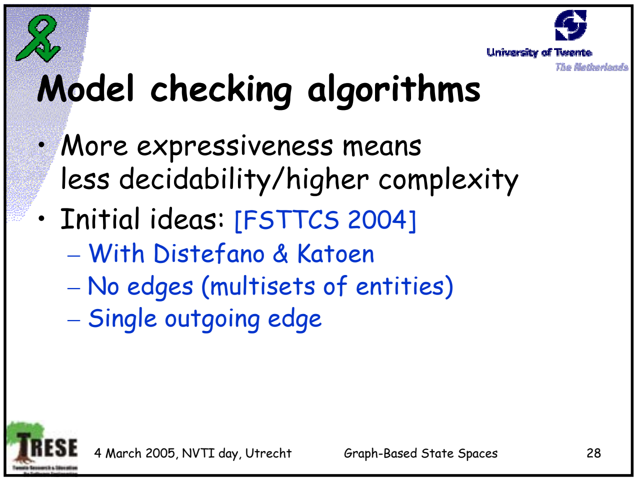


University of Twente
The Netherlands

Issues to be addressed

- Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs) not applicable
- Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- ➔ Propositional logic not suitable
- Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)

TRESE 4 March 2005, NVTI day, Utrecht Graph-Based State Spaces 27

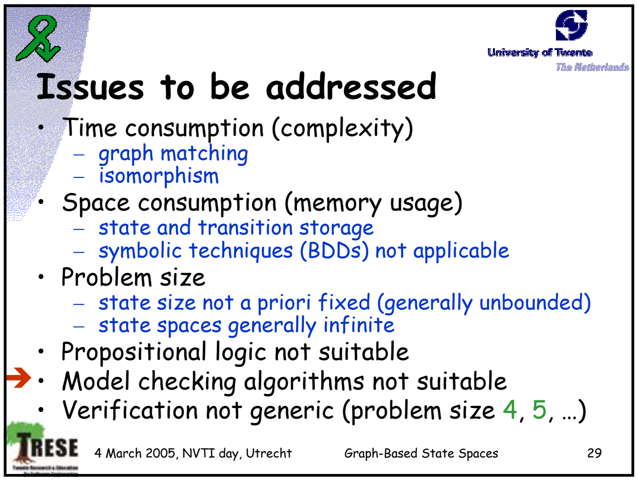


University of Twente
The Netherlands

Model checking algorithms

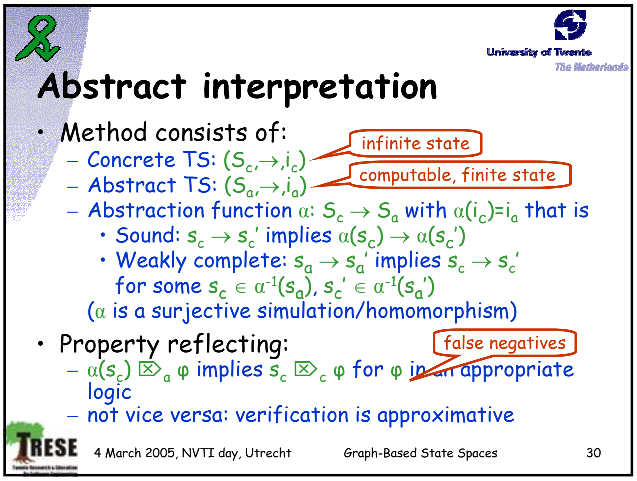
- More expressiveness means
  less decidability/higher complexity
- Initial ideas: [FSTTCS 2004]
  - With Distefano & Katoen
  - No edges (multisets of entities)
  - Single outgoing edge

TRESE 4 March 2005, NVTI day, Utrecht Graph-Based State Spaces 28



Issues to be addressed

- Time consumption (complexity)
  - graph matching
  - isomorphism
- Space consumption (memory usage)
  - state and transition storage
  - symbolic techniques (BDDs) not applicable
- Problem size
  - state size not a priori fixed (generally unbounded)
  - state spaces generally infinite
- Propositional logic not suitable
- ➔ Model checking algorithms not suitable
- Verification not generic (problem size 4, 5, …)

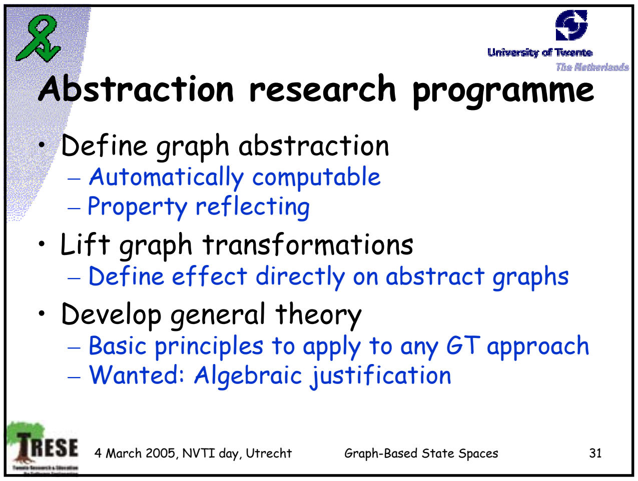


Abstract interpretation

- Method consists of:
  - Concrete TS: $(S_c,\rightarrow,i_c)$ infinite state
  - Abstract TS: $(S_a,\rightarrow,i_a)$ computable, finite state
  - Abstraction function α: $S_c \rightarrow S_a$ with $\alpha(i_c)=i_a$ that is
    - Sound: $s_c \rightarrow s_c'$ implies $\alpha(s_c) \rightarrow \alpha(s_c')$
    - Weakly complete: $s_a \rightarrow s_a'$ implies $s_c \rightarrow s_c'$ for some $s_c \in \alpha^{-1}(s_a)$, $s_c' \in \alpha^{-1}(s_a')$

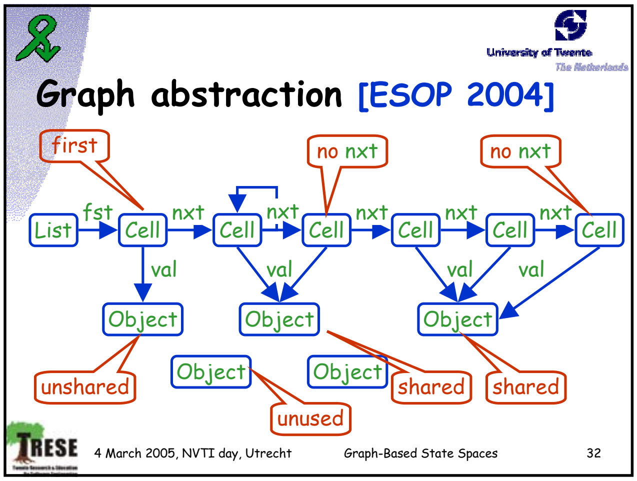
  (α is a surjective simulation/homomorphism)
- Property reflecting:
  - $\alpha(s_c)$ ⌦$_a$ φ implies s_c ⌦$_c$ φ for φ in an appropriate logic
  - not vice versa: verification is approximative

false negatives



Abstraction research programme

- Define graph abstraction
  - Automatically computable
  - Property reflecting
- Lift graph transformations
  - Define effect directly on abstract graphs
- Develop general theory
  - Basic principles to apply to any GT approach
  - Wanted: Algebraic justification



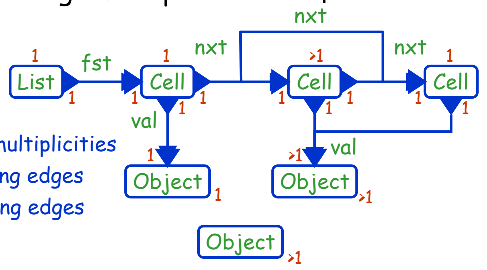
Graph abstraction [ESOP 2004]

Enriching abstract graphs

- The following information is added:
  - The (potential) number of node instances
  - The (potential) degree of sharing (in+out)
- Both can be expressed as multiplicities
- Strongly inspired by *shape graphs*
  - Sagiv, Reps, Wilhelm, Benedikt

Pictorial representation

- Write edge multiplicities at "ports"

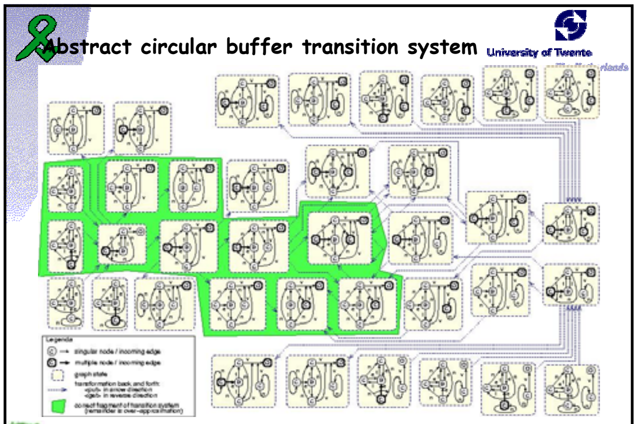


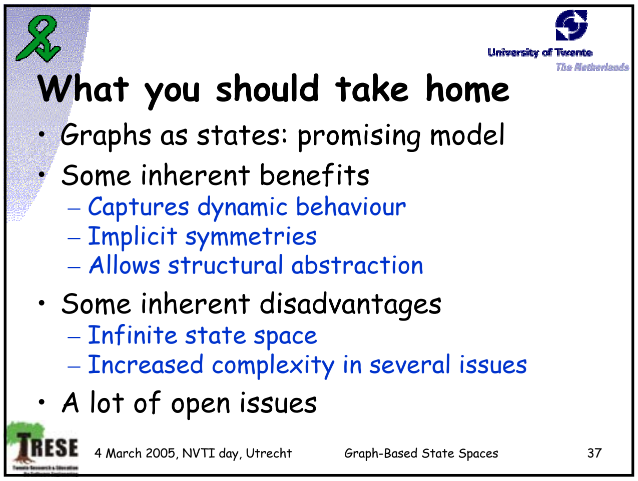
- Node multiplicities
- Outgoing edges
- Incoming edges

Abstract graph transformation

- Materialization
  - Matching of left hand side made concrete
  - Result: partially concrete graph
- Transformation
  - Partially concrete graph treated as fully concrete
- Normalization
  - Transformation result is partially concrete
  - Re-apply abstraction principle

Abstract circular buffer transition system





What you should take home

- Graphs as states: promising model
- Some inherent benefits
  - Captures dynamic behaviour
  - Implicit symmetries
  - Allows structural abstraction
- Some inherent disadvantages
  - Infinite state space
  - Increased complexity in several issues
- A lot of open issues