Regular Model Checking Using Inference of Regular Languages

with Tomas VOJNAR (TU Brno, Czech Republic)

Peter Habermehl (LIAFA, University Paris 7, France)

September 15th, 2004

Introduction

- Regular Model Checking
- Inference (Learning) of Regular Languages
- Using inference for regular model checking
  - General model-checking algorithm
  - Experiments

Regular Model Checking

[Kersten et al. 97, Fribourg et al. 97 (Infinity)]

- Configurations of systems are modeled as strings over a finite alphabet Σ
- Finite automata A over Σ represent (infinite) regular sets of configurations.
  - $Init$: set of initial configurations
  - Bad: set of bad configurations
- Transitions are modeled by a transducer τ (automata over $\Sigma \times \Sigma$).
- Reachable configurations in n steps: $\tau^n(Init)$
- $\tau^*(Init) := \bigcup_{k=0}^{\infty} \tau^k(Init)$ (not necessarily regular)
- $\tau^* := \bigcup_{k=0}^{\infty} \tau^k$ (not necessarily regular)

Regular Model Checking

- Model-checking problem 1: $\tau^*(Init) \cap Bad = \emptyset$?
- Model-checking problem 2: $\tau^* \cap \tau_{Bad} = \emptyset$?
- Several approaches exist
  [Abdulla, Boigelot, Bouajjani, Jonsson, Nilsson, Pnueli, Wolper, etc.]
- Calculating exact reachability sets or relations
  - Special classes where τ^* can be calculated
  - Exact widening
- Calculating overapproximations (which are sometimes exact)
  - Abstract regular model-checking [Bouajjani et al. CAV 04]
  - Inference of regular languages (extending [Fribourg et al. 97])

Regular Model Checking

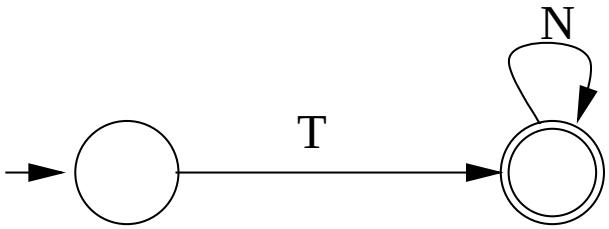
A lot of different systems can be encoded in this way, for example

- (lossy) FIFO-channel systems
- pushdown systems
- systems with counters
- parameterized systems (parameterized number of identical finite-state processes arranged in an array)

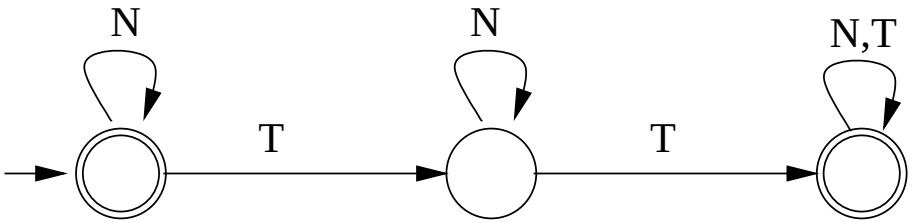
Example

- Very simple token ring protocol
- Processes are arranged in a linear array.
- An individual process has (T) or has not (N) the token.
- A string of $\{N, T\}^*$ of arbitrary length represents a configuration.
- The token can be passed from left to right
  ($NTNN \rightarrow NNTN$, $NTN \rightarrow NNT$).

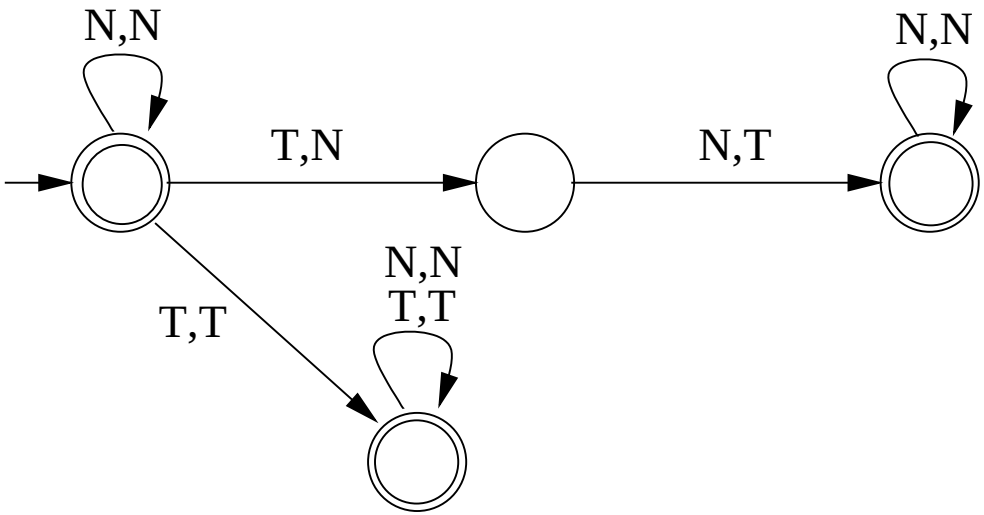
Example: Token Ring



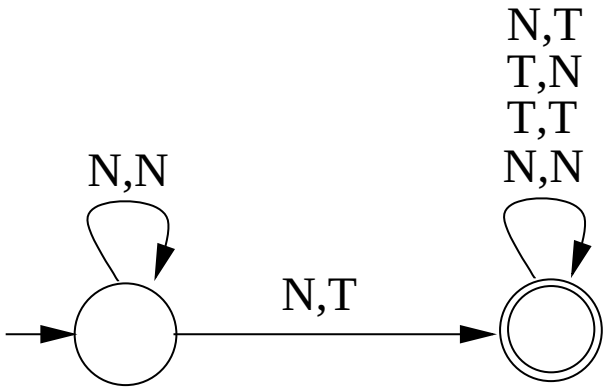
Initial configurations



Bad configurations

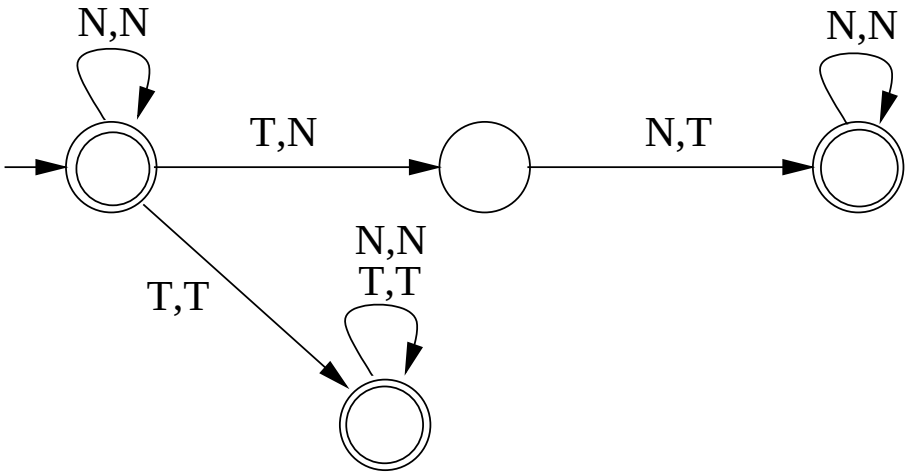


Transitions encoded by transducer τ

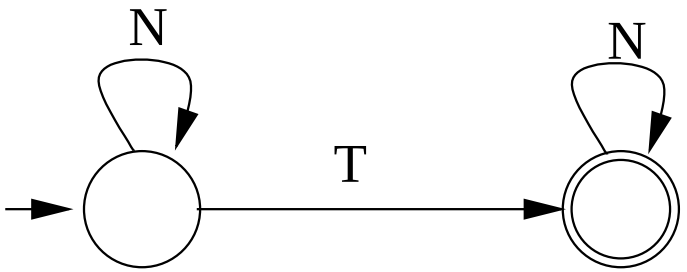


τ_{Bad}

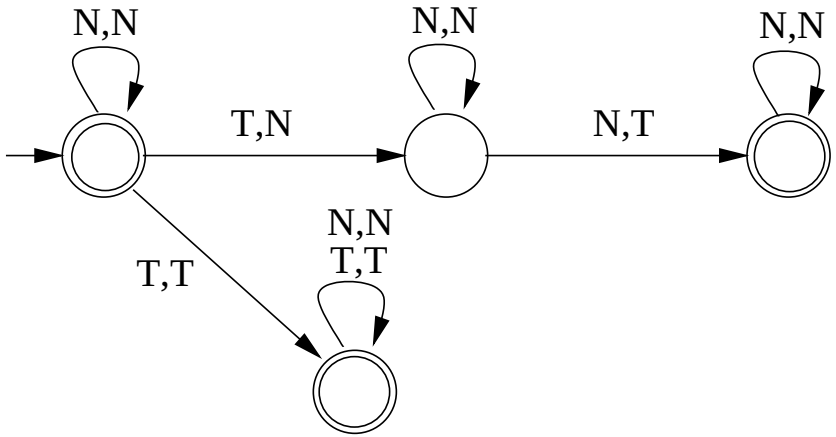
Example



τ



$\tau^*(Init)$



τ^*

Length-preserving transducers

- Transducers do not contain ϵ.
- For safety (reachability) properties this is not a restriction.
- Key observation for length-preserving transducers:
  - $\tau^*(Init)$ restricted to configurations of bounded size can be computed.
  - τ^* restricted to bounded lengths can be computed.
  - These finite sets can be considered complete samples of $\tau^*(Init)$ and τ^* resp.
  - gives rise to a special inference problem

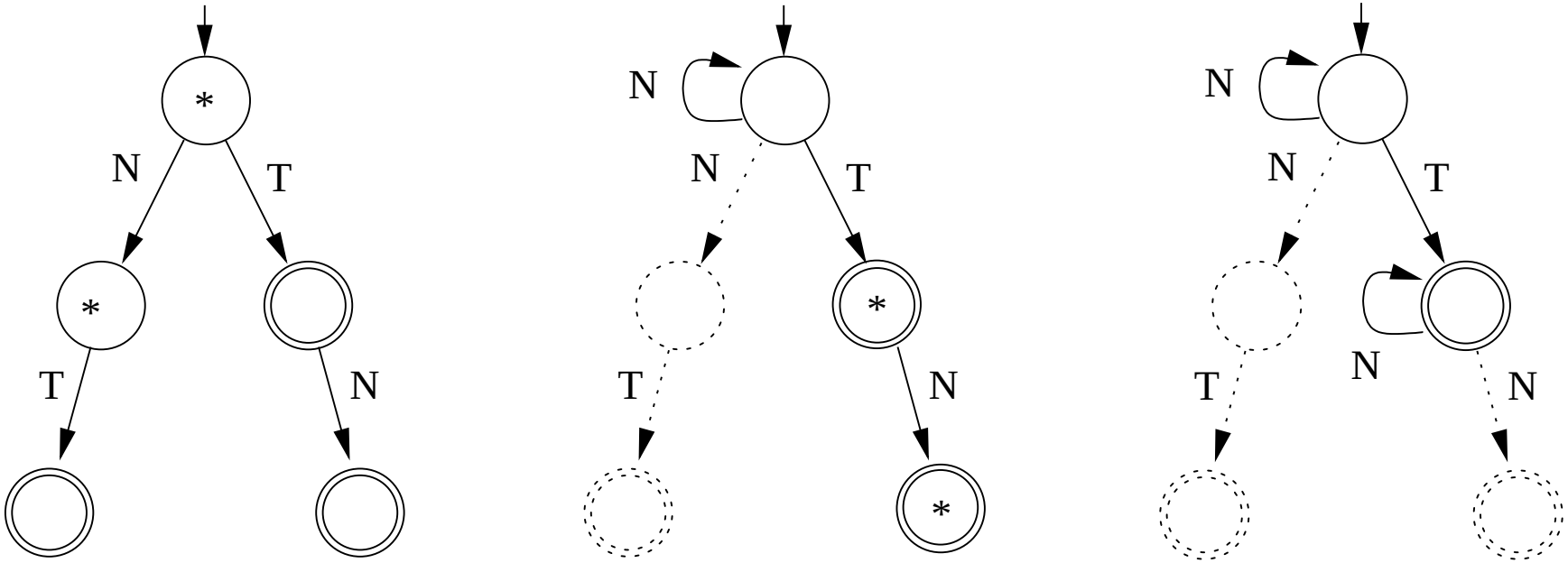
Inference of regular languages from complete training sets

[Trakhtenbrot, Barzdin 72]

- Automata to be infered: A
- Complete training set $T_k = (T_k^+, T_k^-)$, where T_k^+ contains all words of $L(A)$ up to length k and T_k^- all others.
- Construct prefix-tree automaton from T_k^+.
- Collapse compatible states (which do not introduce inconsistencies)
- Theorem: given a sufficiently big (depends on the structure of the automaton) complete training set the Trakhtenbrot-Barzdin algorithm infers A.
  In the worst case all words of $L(A)$ up to length $2|A| - 1$ are needed.

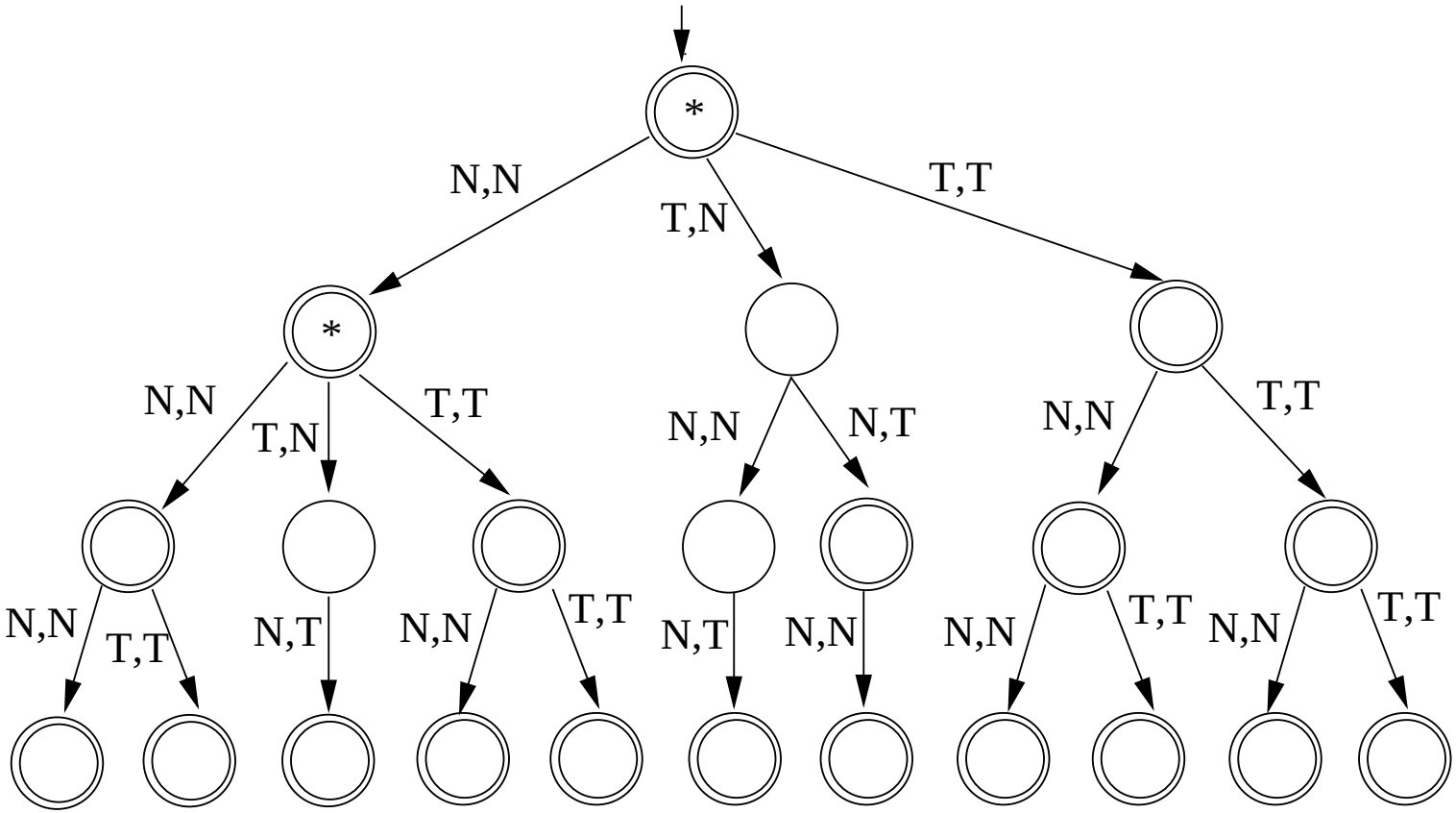
Example

Inference of $\tau^*(Init)$ from a complete sample of size 2.



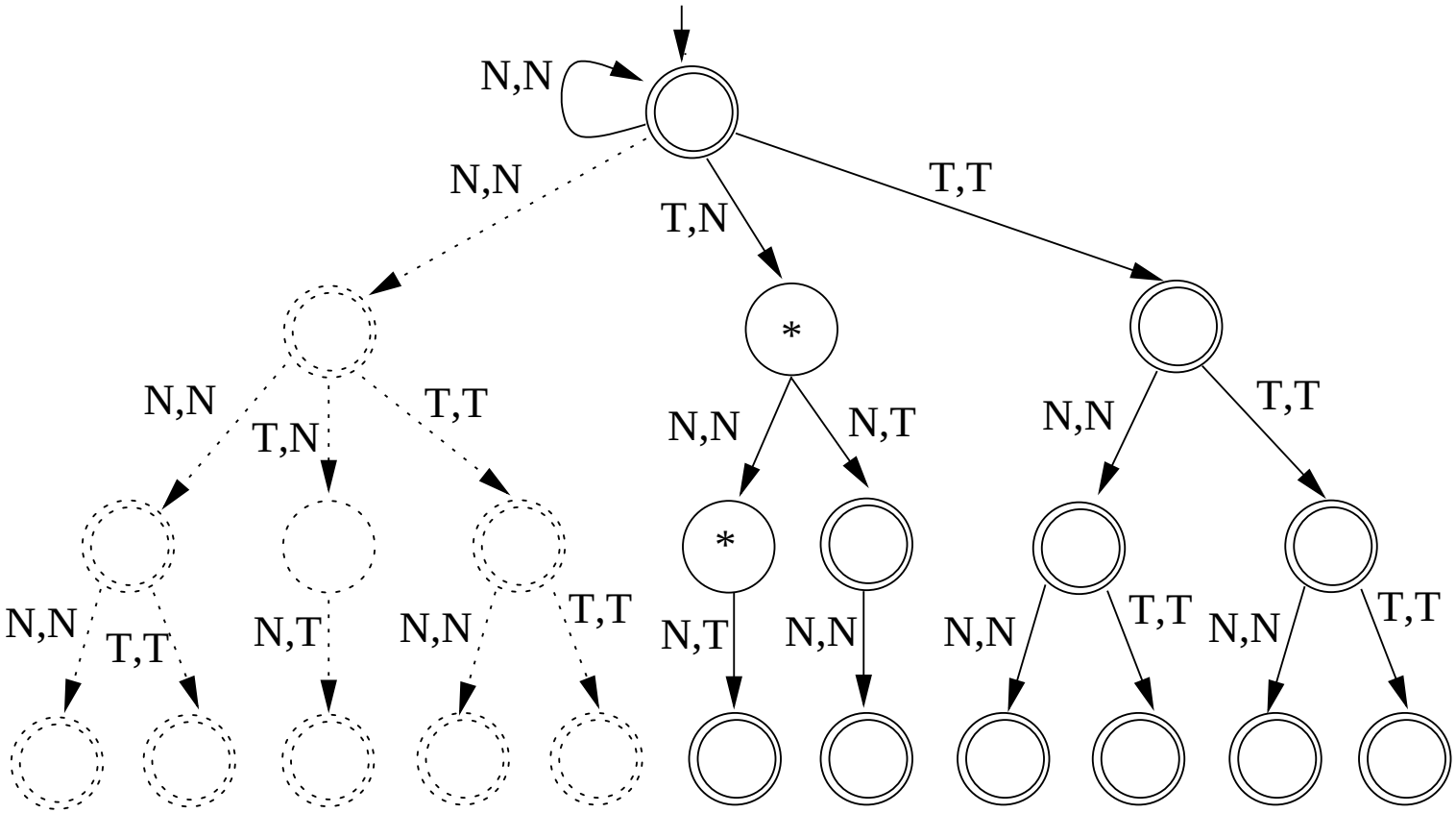
Example

Inference of an overapproximation of τ^* from a complete sample of size 3



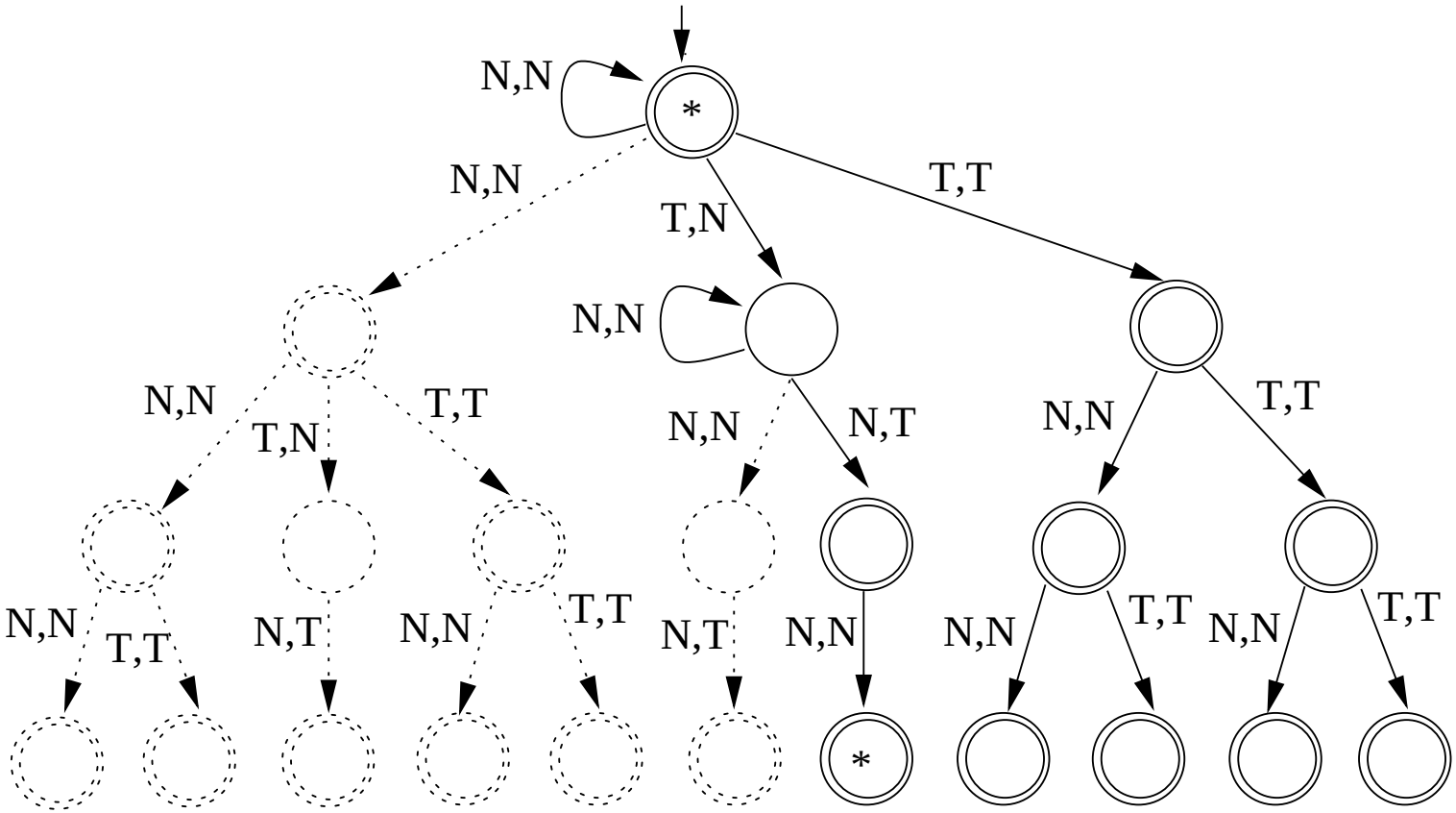
Example

Inference of an overapproximation of τ^* from a complete sample of size 3



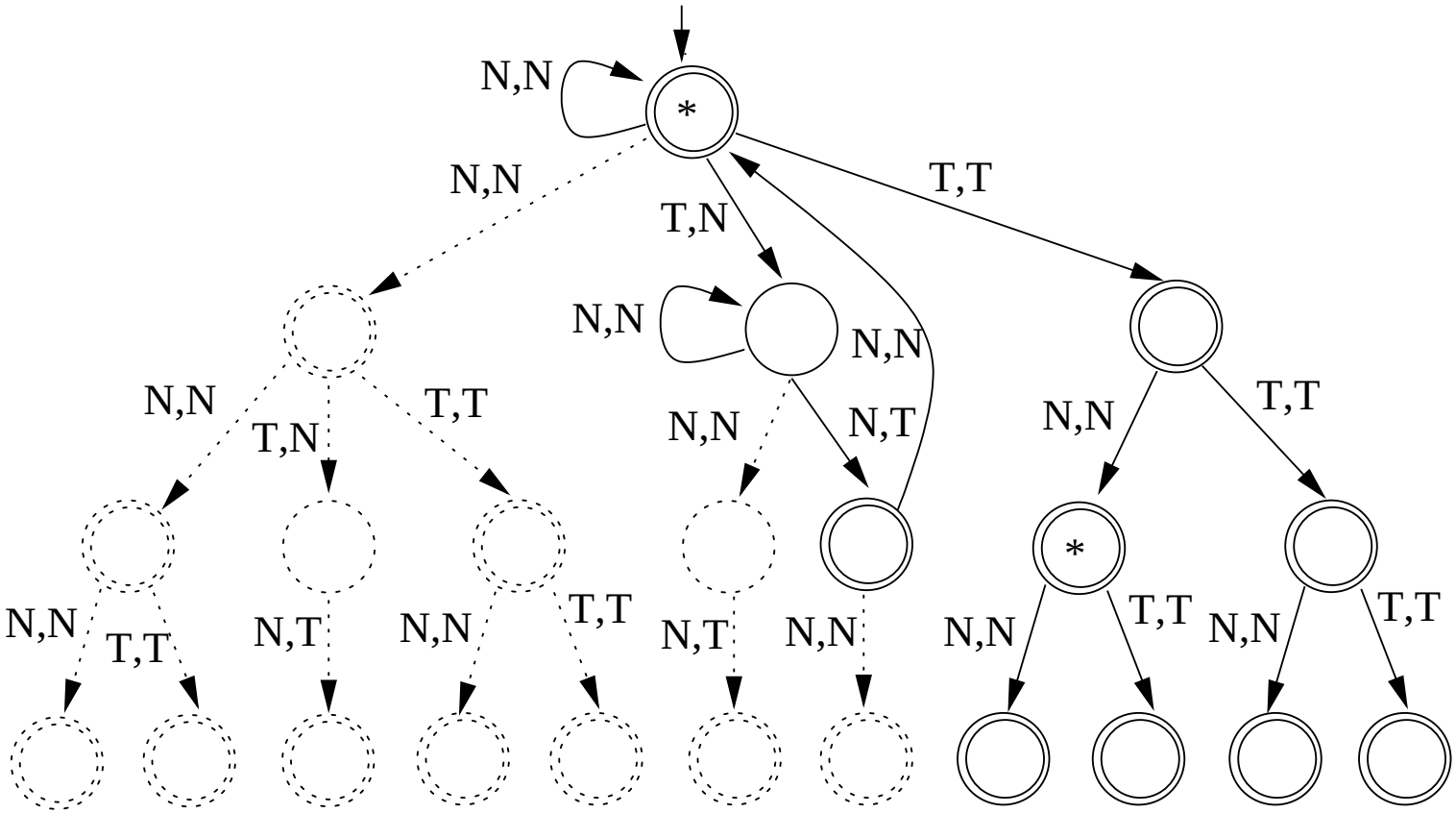
Example

Inference of an overapproximation of τ^* from a complete sample of size 3



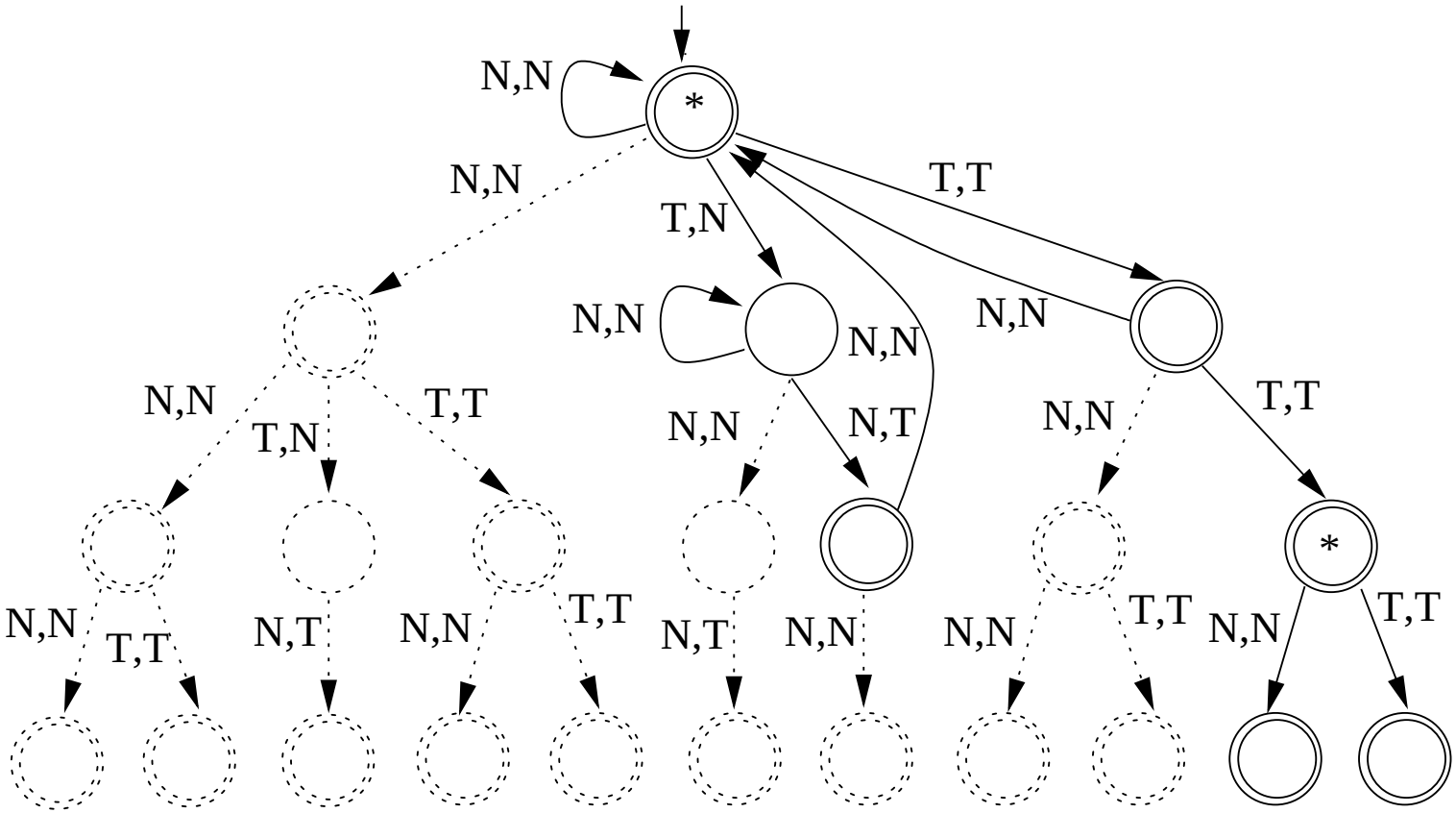
Example

Inference of an overapproximation of τ^* from a complete sample of size 3



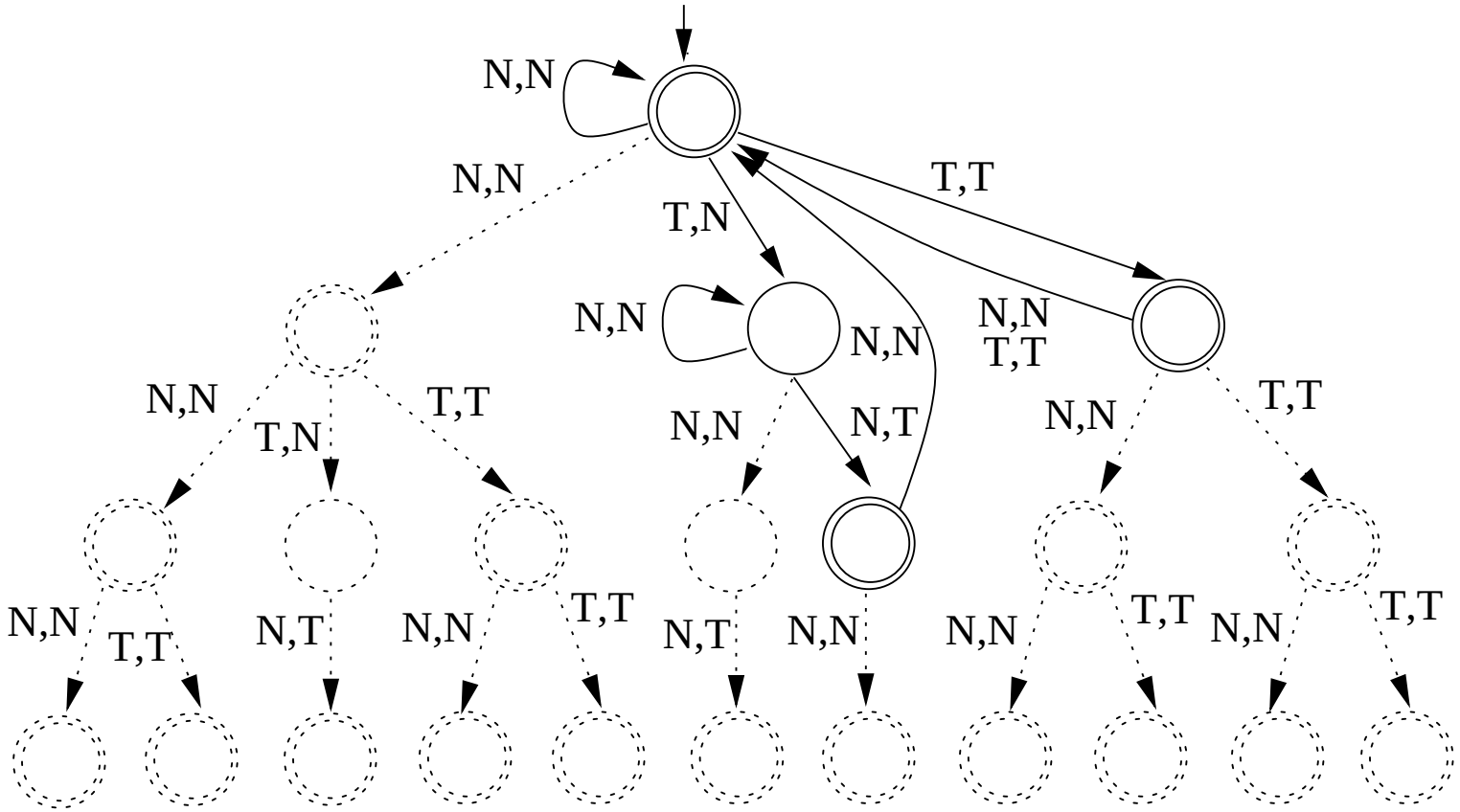
Example

Inference of an overapproximation of τ^* from a complete sample of size 3



Example

Inference of an overapproximation of τ^* from a complete sample of size 3



A general model-checking algorithm

Problem 1: $\tau^*(Init) \cap Bad = \emptyset$?

`input:` a length-preserving transducer τ,
a regular set of initial configurations $Init$
and a regular set of bad configurations Bad
$i := 1$; /* i can be initialised differently too. */
`repeat`
$C := \tau^*(Init^{\leq i})$;
$\overline{C} := \Sigma^{\leq i} \setminus C$;
`if` $Bad \cap C \neq \emptyset$ `then output:` property violated;
$A := inference(C, \overline{C})$;
$i := i + 1$;
`until` $\tau(L(A)) \subseteq L(A)$ and $Init \subseteq L(A)$ and $L(A) \cap Bad = \emptyset$;
`output:` property satisfied

As inference algorithm one can use for example Trakhtenbrot-Barzdin

Results

- The model-checking algorithm always terminates if $\tau^*(Init)$ is regular.
- Model-checking problem 1 is decidable if $\tau^*(Init)$ is regular.
- This already follows from [Pachl 87]. It is sufficient to enumerate all regular languages and check them for invariance. Here we provide a clever enumeration.
- The algorithm can be easily used for τ^* as well.
- Model-checking problem 2 is decidable if τ^* is regular.

Experiments

Prototype implementation in Prolog using FSA Library

Experiment	T[sec]	G[%]	Experiment	T[sec]	G[%]
Bakery	0.03	50	Dijkstra	1.16	92
Bakery comm. liv.	0.36	90	PDS	0.04	63
Bakery counters 3P	8.69	70	Petri net/Read. Wr.	323	90
Bakery counters 4P	143	92	Faulty PN/Rd. Wr.	1.48	54
Bakery 5P unary	229	45	Szymanski	0.76	94
ABP	0.03	50	Rev. Lists	1.64	90
Burns	0.77	98	Rev. Lists/Transd.	40.5	69

Conclusion and Perspectives

- General algorithm for Regular model-checking
- Termination guaranteed for regular reachability sets
- Try other inference algorithms
- Use dedicate algorithms for generating reachable configurations of bounded length.