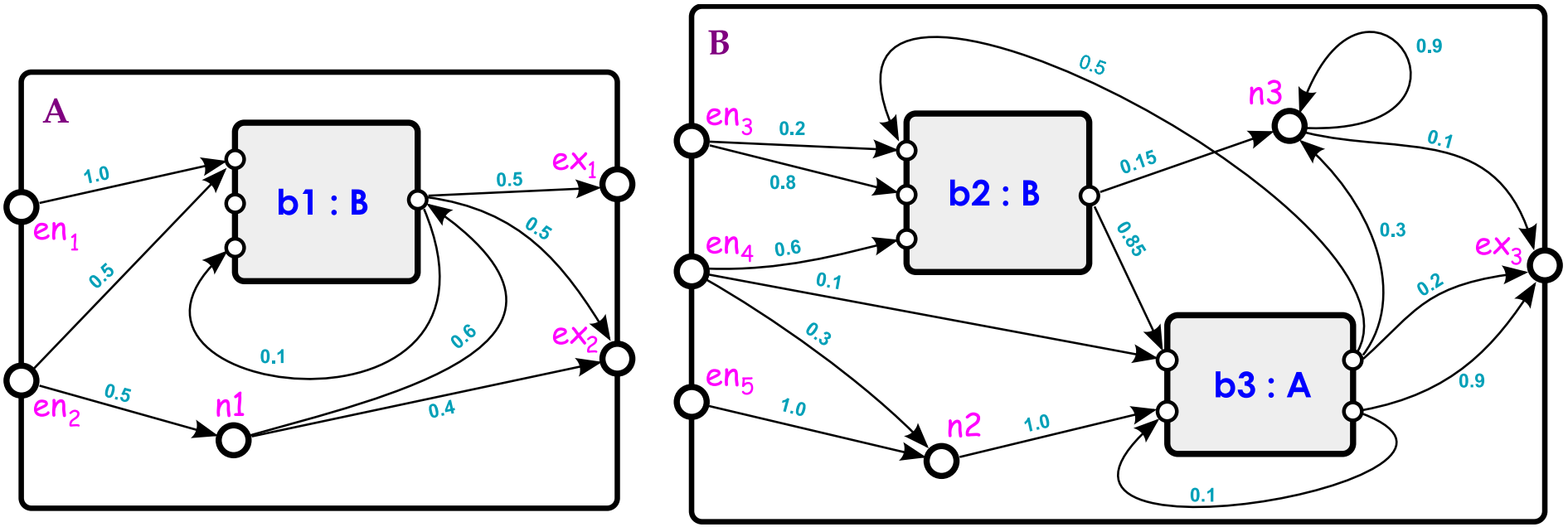
# PReMo:
## An analyzer for Probabilistic Recursive Models

Dominik Wojtczak

School of Informatics, University of Edinburgh

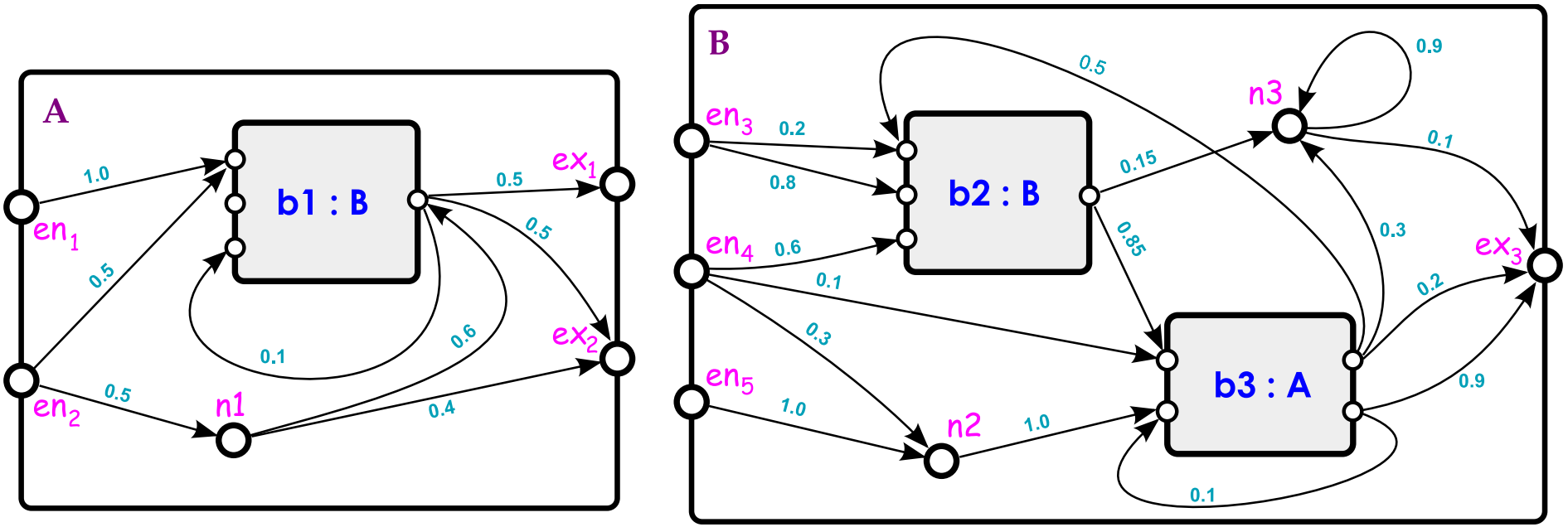
Two Decades of Probabilistic Verification
Leiden, Netherlands
12th-16th November 2007

# What is a Recursive Markov Chain(RMC)?




- possible infinite number of states
- in a natural way captures imperative function calls
- probability of termination
- general model checking questions

# What is a Recursive Markov Chain(RMC)?




- possible infinite number of states
- in a natural way captures imperative function calls
- probability of termination
- general model checking questions

# Related models

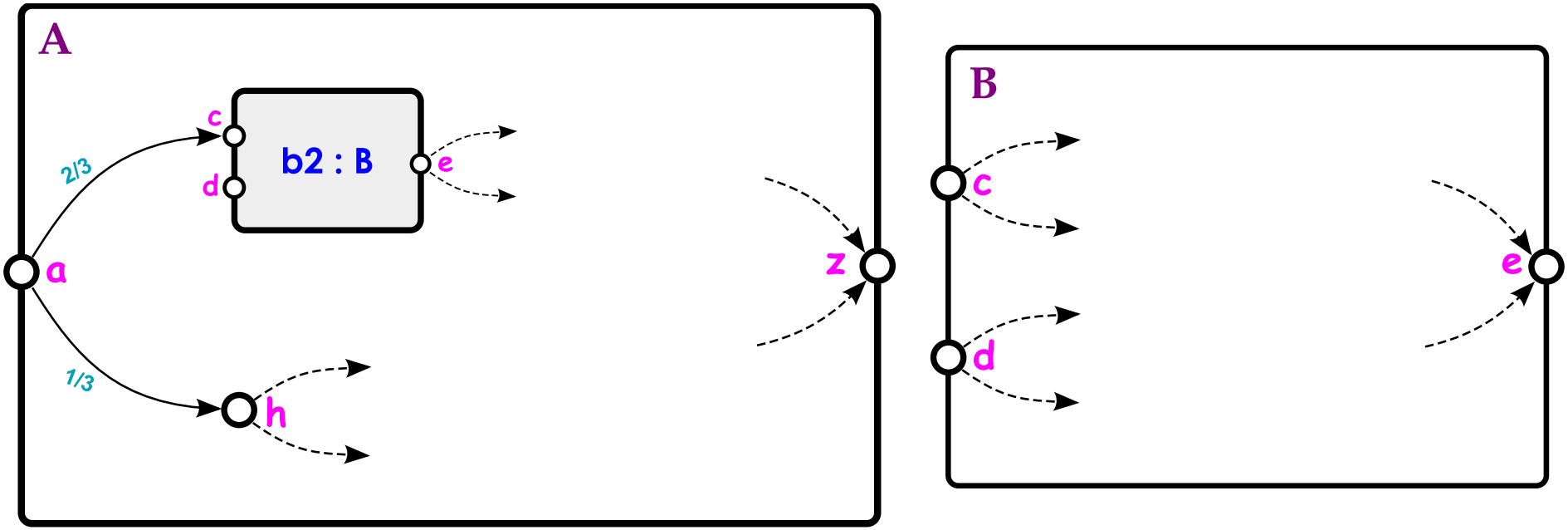
## Subsumes

- Recursive State Machines / Pushdown Systems / Boolean Programs (used in analyzing imperative programs with recursion)
- finite Markov Chains
- Stochastic Context-Free Grammars (used in NLP)
- Multi-Type Branching Processes (population dynamics)
- the two previous ones are embeddable into 1-RMCs
- Quasi-Birth-Death Processes (discrete time) [EY07]

## Equivalent models

- probabilistic Pushdown Automata
- as expressive as discrete time tree-like and tree structured QBDs [EY07]

# How can we compute the termination probability in an RMC?




- exit z: $x_{A,z} = 1$
- node a: $x_{A,a} = \frac{1}{3}x_{A,h} + \frac{2}{3}x_{A,c}$
- box b1 entry c: $x_{A,c} = x_{B,c}x_{A,d}$
- do it for every single node and get a system of equations $\mathbf{x} = \mathbf{P}(\mathbf{x})$
- operator P is monotone and its Least Fixed Point gives us the termination probabilities

# Simple methods for computing the Least Fixed Point of **P**

## Jacobi standard iteration

$$\mathbf{x}^k = \mathbf{P}(\mathbf{x}^{k-1})$$

## Gauss-Seidel iteration

$$\mathbf{x}_i^k := \mathbf{P}_i(\mathbf{x}_1^k, \ldots, \mathbf{x}_{i-1}^k, \mathbf{x}_i^{k-1}, \mathbf{x}_{i+1}^{k-1}, \ldots, \mathbf{x}_n^{k-1})$$

## Successive Over Relaxation(SOR) iteration

an “optimistic” modification of Gauss-Seidel that might not converge

there are simple examples for which all these algorithms need $2^i$ steps or more to obtain $i$-bits of precision!

# Multivariate Newton's method

Objective: given $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ find $\mathbf{x}$ such that $\mathbf{F}(\mathbf{x}) = \mathbf{0}$
$\mathbf{x}^0 = \mathbf{c}$
$\mathbf{x}^{k+1} = \mathbf{x}^k - (\mathbf{F}'(\mathbf{x}^k))^{-1} \cdot \mathbf{F}(\mathbf{x}^k)$
where $\mathbf{F}'(\mathbf{x})$ is a Jacobian matrix

Run for $\mathbf{F}(\mathbf{x}) = \mathbf{P}(\mathbf{x}) - \mathbf{x}$ and $\mathbf{c} = \mathbf{0}$ after some SCC decomposition.

## Dense Newton

in $\mathbf{x}^{k+1} := \mathbf{x}^k - (\mathbf{F}'(\mathbf{x}^k))^{-1}\mathbf{F}(\mathbf{x}^k)$

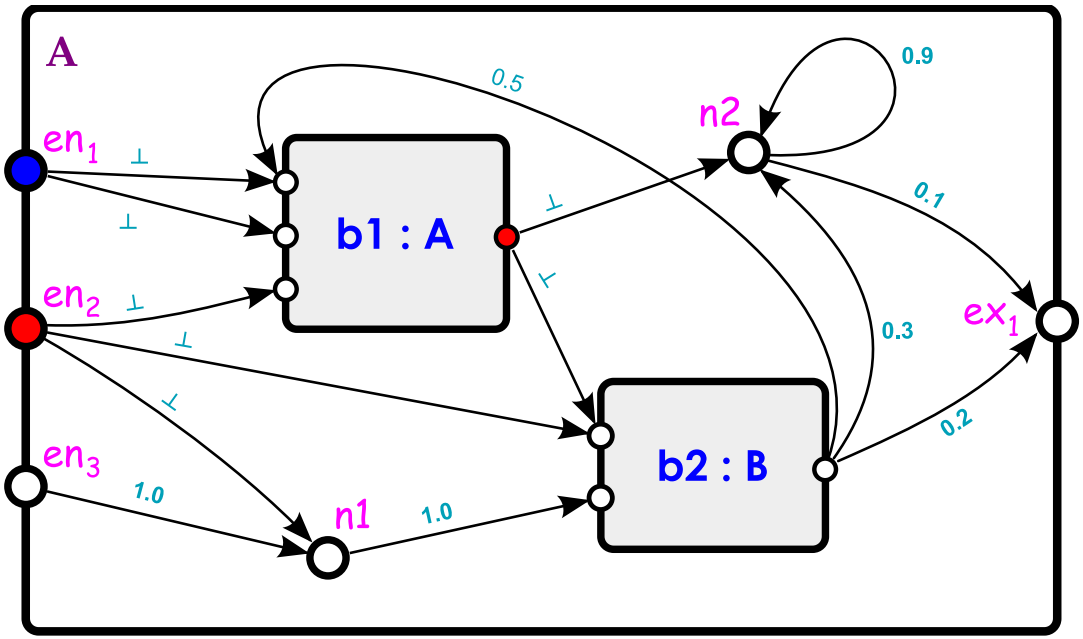
- compute $(\mathbf{F}'(\mathbf{x}^k))^{-1}$ directly through LU decomposition

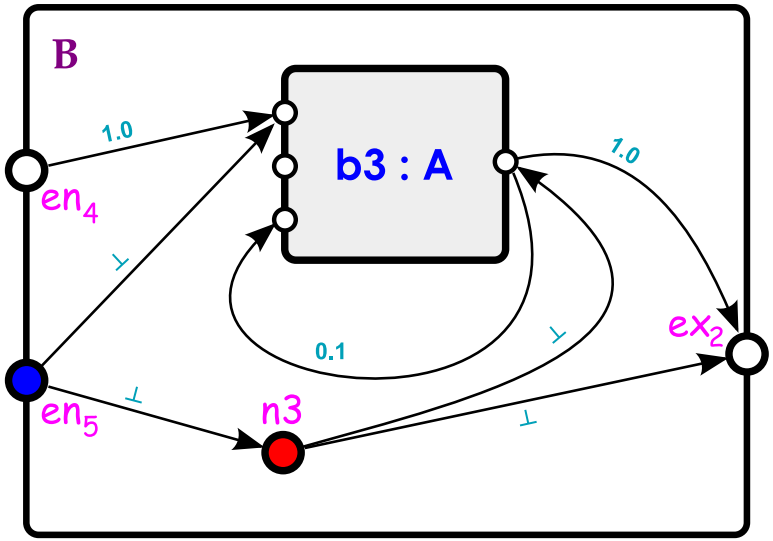
## Sparse Newton

transform the equation into $(\mathbf{F}'(\mathbf{x}^k))(\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{F}(\mathbf{x}^k)$

- solve this equation using sparse linear solver (5 to choose from)

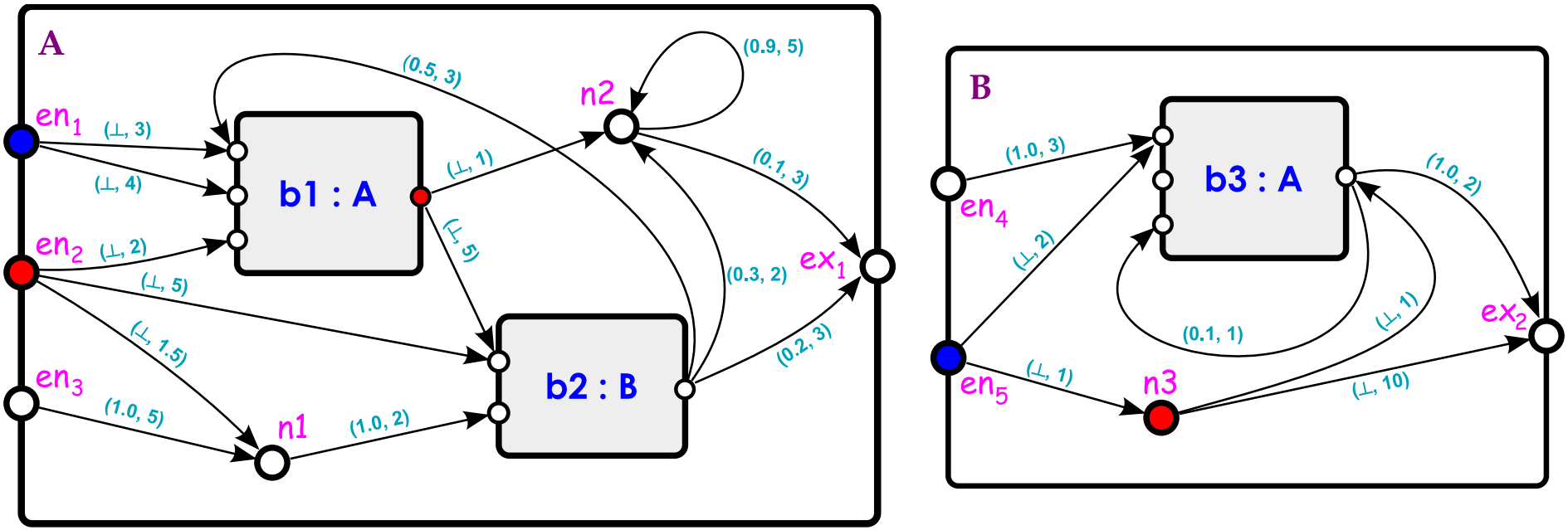
# Recursive MDPs and Recursive Stochastic Games




- adding controller(s) to the program
- each component has just one exit

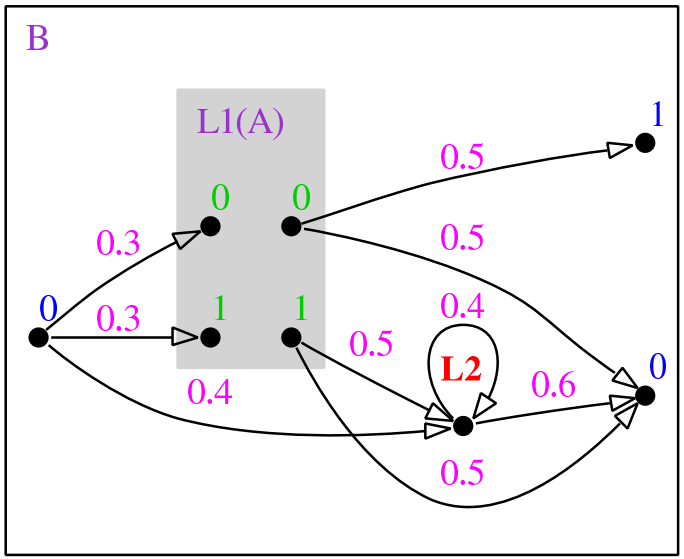
# Recursive MDPs and Recursive Stochastic Games with rewards

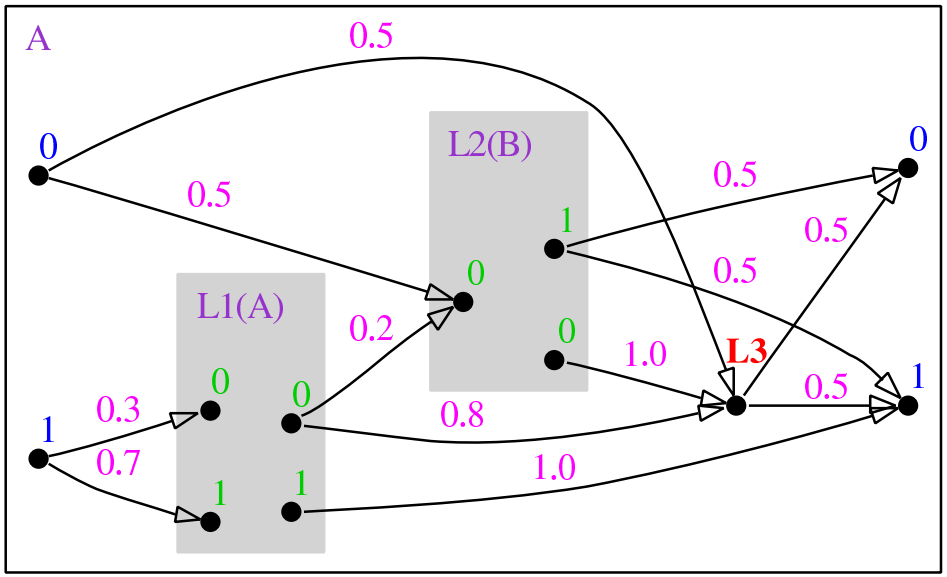



- all rewards are positive and each component has just one exit
- analyzing pessimal/best/expected running time of a program
- for RMCs analyzing variance, average reward per transition etc. (not yet in PReMo)

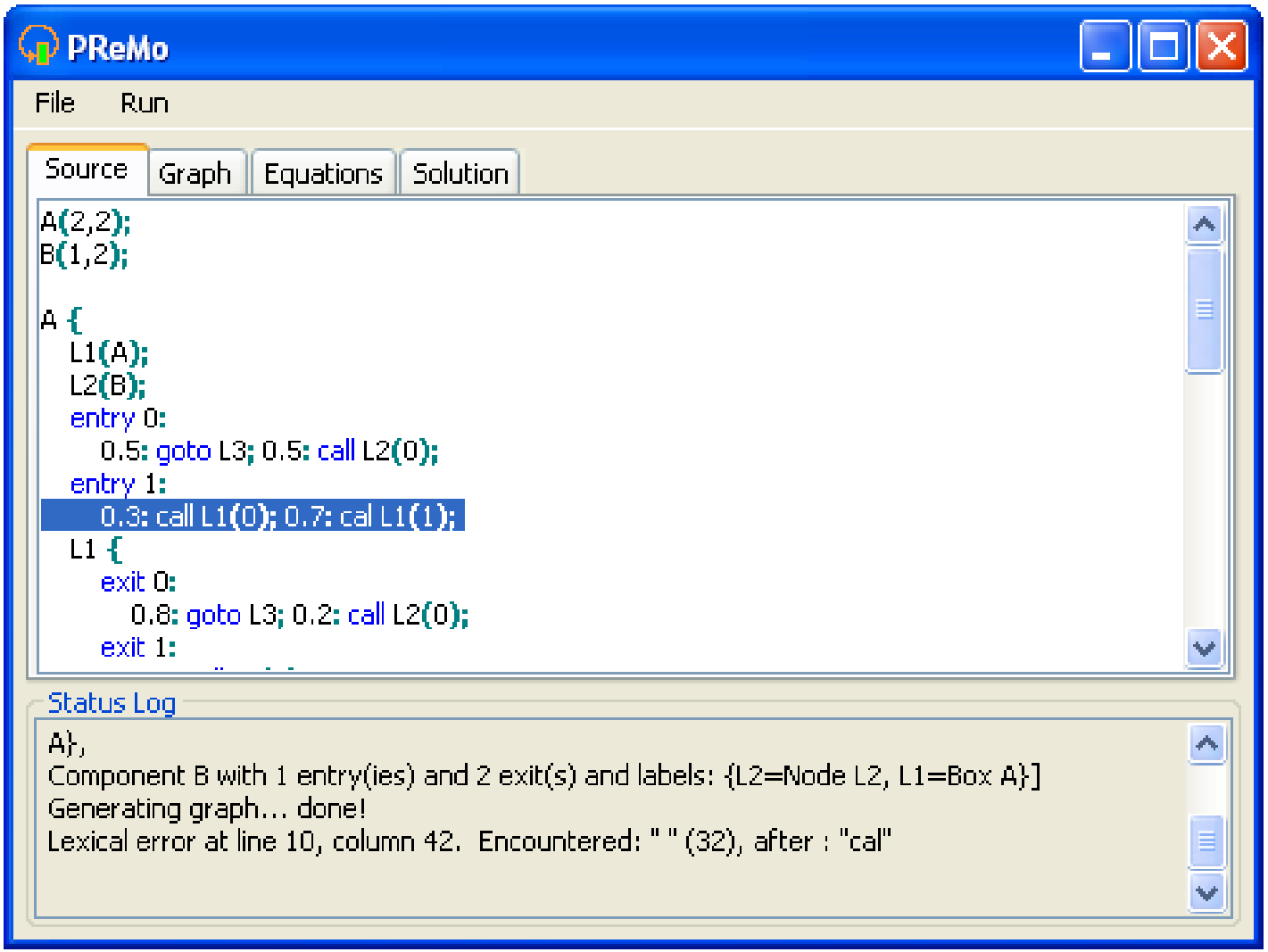
# Input language and graphical depiction

```
A(2,2);
B(1,2);

A {
  L1(A);
  L2(B);
  entry 0:
    0.5: goto L3; 0.5: call L2(0);
  entry 1:
    0.3: call L1(0); 0.7: call L1(1);
  L1 {
    exit 0:
      0.8: goto L3; 0.2: call L2(0);
    exit 1:
      1.0: return 1;
  }
  L2 {
    exit 0:
      1.0: goto L3;
    exit 1:
      0.5: return 0; 0.5: return 1;
  }
  L3 {
    0.5: return 0; 0.5: return 1;
  }
}

B {
  L1(A);
  entry 0:
    0.3: call L1(0); 0.3: call L1(1); 0.4: goto L2;
  L1 {
    exit 0:
      0.5: return 0; 0.5: return 1;
    exit 1:
      0.5: return 0; 0.5: goto L2;
  }
  L2 {
    0.4: goto L2; 0.6: return 0;
  }
}
```



# PReMo's user interface




- simple "imperative-style" format
- syntax highlighting
- errors detection
- auto-indentation

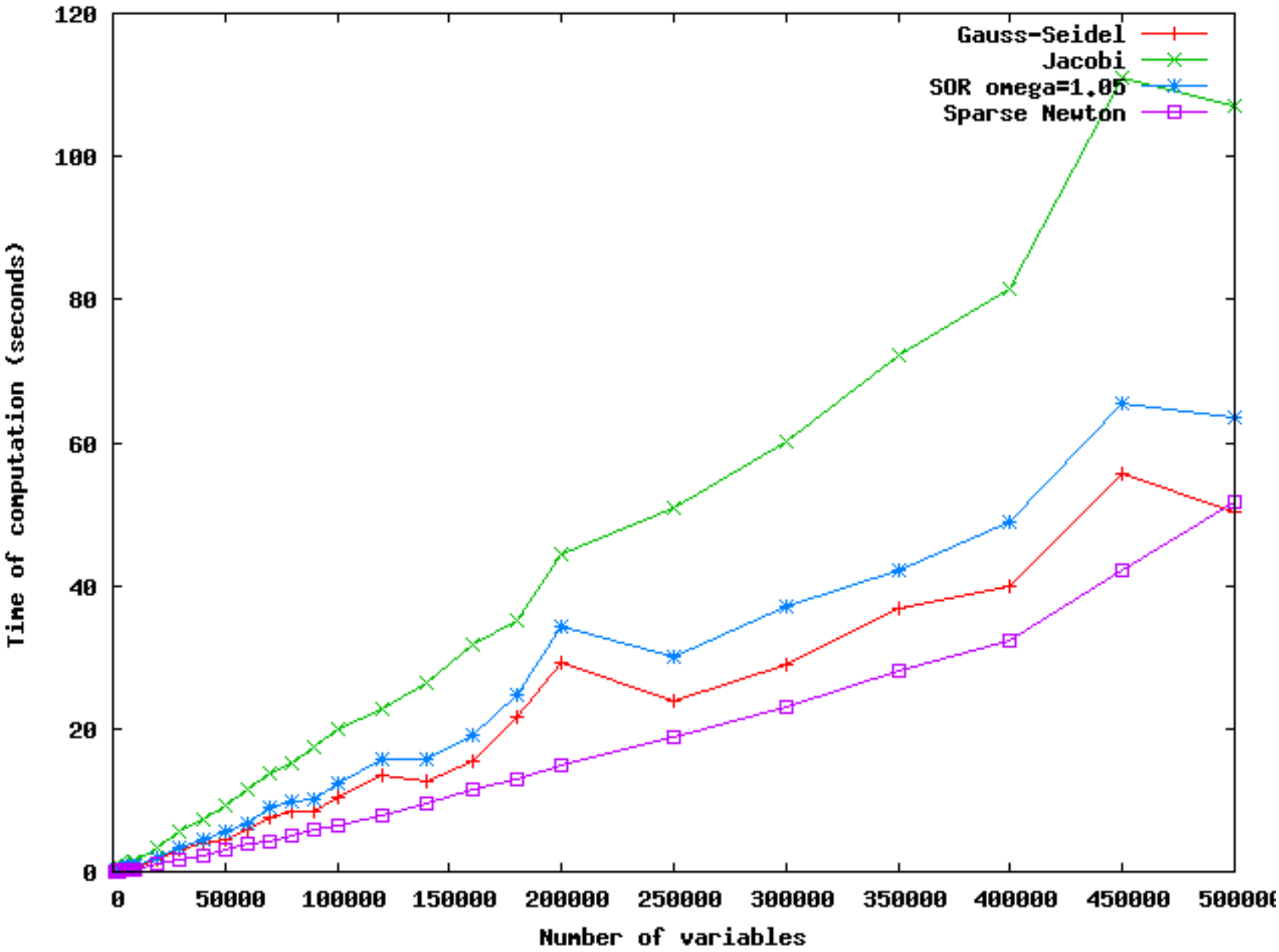
# Analyzing SCFGs from the Penn Treebank corpora

## Definition

Stochastic CFG is consistent iff it generates a finite word with probability 1

| name | #prod | max-scc | Jacobi | Gauss Seidel | SOR $\omega$=1.05 | DNewton | SNewton |
|---|---|---|---|---|---|---|---|
| brown | 22866 ✘ | 448 | 312.084(9277) | 275.624(7866) | diverge | 2.106(8) | 2.115(9) |
| lemonde | 32885 ✓ | 527 | 234.715(5995) | 30.420(767) | diverge | 1.556(7) | 2.037(7) |
| negra | 29297 ✓ | 518 | 16.995(610) | 4.724(174) | 4.201(152) | 1.017(6) | 0.499(6) |
| swbd | 47578 ✘ | 1123 | 445.120(4778) | 19.321(202) | 25.654(270) | 6.435(6) | 3.978(6) |
| tiger | 52184 ✓ | 1173 | 99.286(1347) | 16.073(210) | 12.447(166) | 5.274(6) | 1.871(6) |
| tuebadz | 8932 ✓ | 293 | 6.894(465) | 1.925(133) | 6.878(461) | 0.477(7) | 0.341(7) |
| wsj | 31170 ✓ | 765 | 462.378(9787) | 68.650(1439) | diverge | 2.363(7) | 3.616(8) |

# Randomly Generated Examples



# Conclusion

## Things already implemented

- support for inputting RMCs, Recursive MDPs and SCFGs
- computing termination probability and expected termination time
- generation of graphical representation of the model

## Possible extensions

- model checking of RMCs (challenging numerical problems)
- extend the input language to allow variables (Boolean Programs)

# An abstraction of Quicksort in the new language

```
void quicksort(int n) {
        [n]skip; // this simulates the call of the partition function, reward in []
        if[] // branches with guards, specify player in []=rand
        ::  n>1 ->
                if[max]
                for i=1..n-1 // macro expansion
                    :1.0:  true -> quicksort(i); quicksort(n-i);
                rof
                fi
        ::  else -> return;
        fi
}

void main() { // every program starts with calling main()
        quicksort(10);
}
```

Compute: reward(main() -> return)

# For More Details

- PReMo's homepage
  http://groups.inf.ed.ac.uk/premo
  (downloadable version for each of the 3 main OSes, user manual and example models)

D. Wojtczak and K. Etessami.
PReMo : An Analyzer for Probabilistic Recursive Models.
In *Proc. of TACAS'07*.

K. Etessami, M. Yannakakis.
A note on the close relationships between variants of probabilitic Pushdown Systems and variants of Quasi-Birth-Death Processes.
Unpublished Draft (in preparation).

THANK YOU!

ANY QUESTIONS?

# For More Details

- PReMo's homepage
  http://groups.inf.ed.ac.uk/premo
  (downloadable version for each of the 3 main OSes, user manual and example models)

D. Wojtczak and K. Etessami.
PReMo : An Analyzer for Probabilistic Recursive Models.
In *Proc. of TACAS'07*.

K. Etessami, M. Yannakakis.
A note on the close relationships between variants of probabilitic Pushdown Systems and variants of Quasi-Birth-Death Processes.
Unpublished Draft (in preparation).

THANK YOU!

ANY QUESTIONS?