Deep Random Search for Efficient Model Checking of Timed Automata

Radu Grosu

Stony Brook University

Joint work with:

X. Huang, S.A. Smolka, W. Tan and S. Tripakis

Embedded Software Systems

- **Difficult to develop & maintain:**
  - **Concurrent and distributed (OS, ES, middleware),**
  - **Complicated by DS improving performance (locks, RC,...),**
  - **Mostly written in C programming language.**
- **Have to be high-confidence:**
  - **Provide the critical infrastructure for all applications,**
  - **Failures are very costly (business, reputation),**
  - **Have to protect against cyber-attacks.**

What is High-Confidence?

Ability to guarantee that

$$S \overset{?}{\models} \varphi$$

system-software S satisfies temporal property φ

Temporal Properties

- **Safety (something bad never happens):**
  - Airborne planes are at least 1 mile apart
  - Nuclear reactor core never overheats
  - Gamma knife never exceeds prescribed dose

- **Liveness (something good eventually happens):**
  - Core eventually reaches nominal temperature
  - Dishwasher tank is eventually full
  - Airbag inflates within 5ms of collision

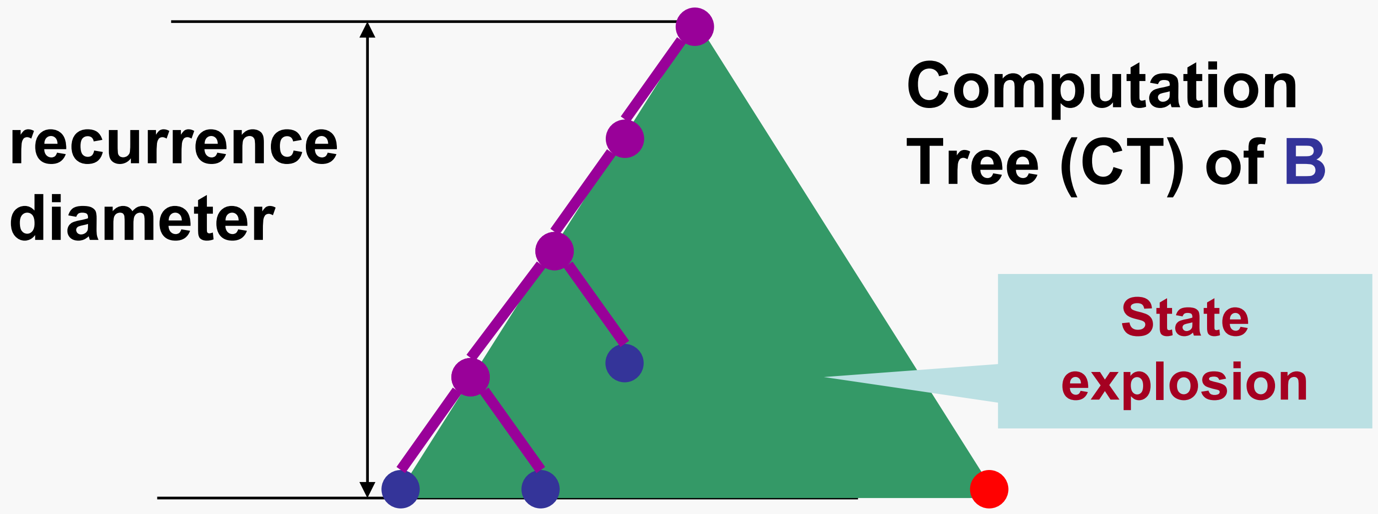
Automata-Theoretic Approach to SP

- Every safety formula φ can be translated to a finite automaton $A_{\neg\varphi}$ such that $L(\neg\varphi) = L(A_{\neg\varphi})$.
- State transition graph of S can also be viewed as a finite automaton A_S (with all states accepting).
- Satisfaction is reduced to language emptiness:

$$S \models \varphi \quad \cong \quad L(A_S) \subseteq L(A_\varphi) \quad \cong \quad L(A_S \times A_{\neg\varphi}) = \varnothing$$

- Language emptiness is reduced to reachability:
  is an accepting state reachable from an initial state?

Checking Non-Emptiness: DFS

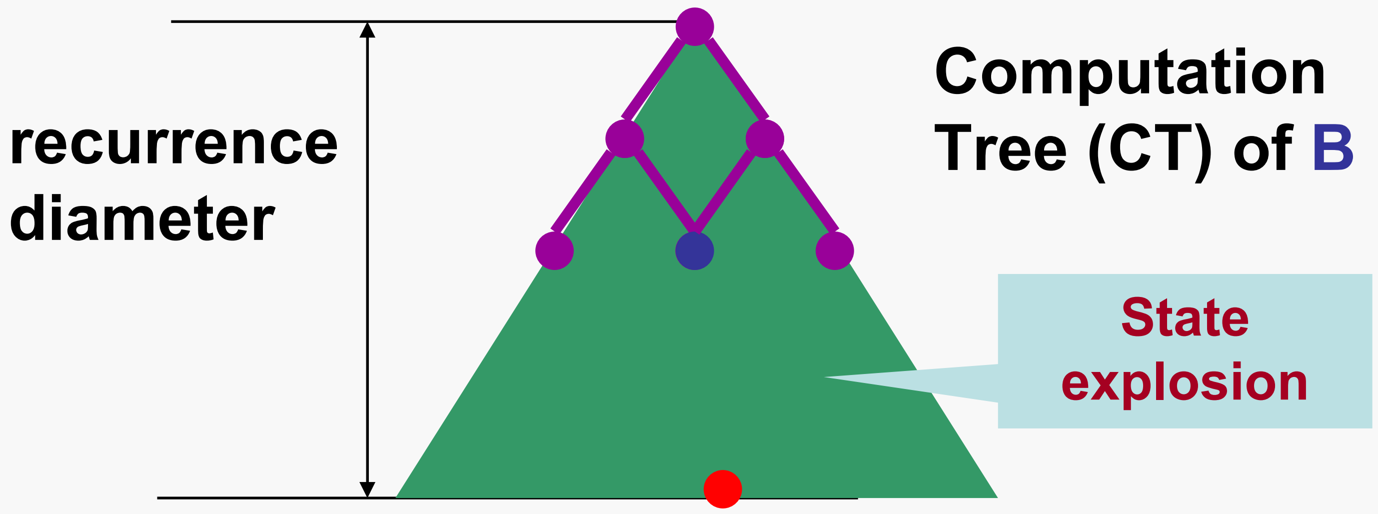


Explore and all reachable states in the CT

Save all states
time efficient

Save current path
memory efficient

Checking Non-Emptiness: BFS



Explore and all reachable states in the CT

Save all states: time efficient

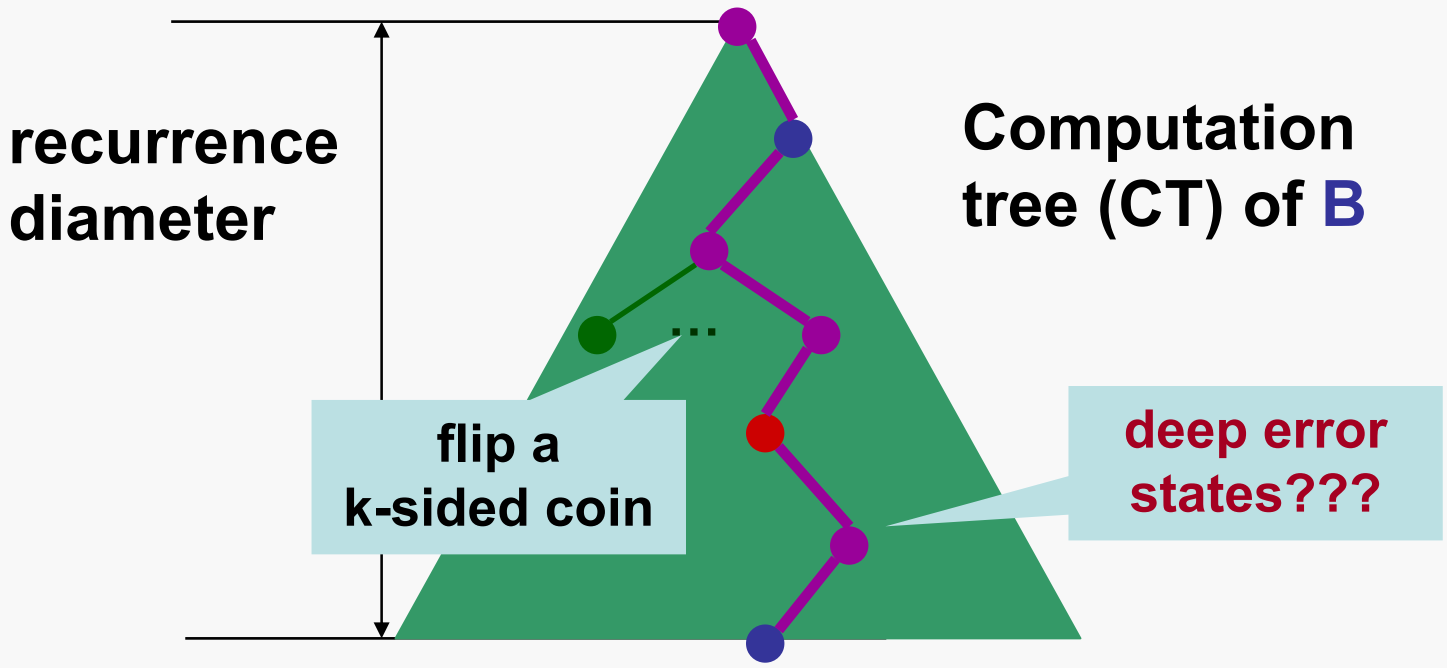
Randomized Algorithms

- **Huge impact on CS:** (distributed) algorithms, complexity theory, cryptography, etc.
- **Takes of next step algorithm** may depend on random choice (coin flip).
- **Benefits of randomization** include simplicity, efficiency, and symmetry breaking.

Randomized Algorithms

- **Monte Carlo:** may produce incorrect result but with bounded error probability.
  - Example: Election's result prediction
- **Las Vegas:** always gives correct result but running time is a random variable.
  - Example: Randomized Quick Sort

Monte Carlo Approach
[TACAS'05]

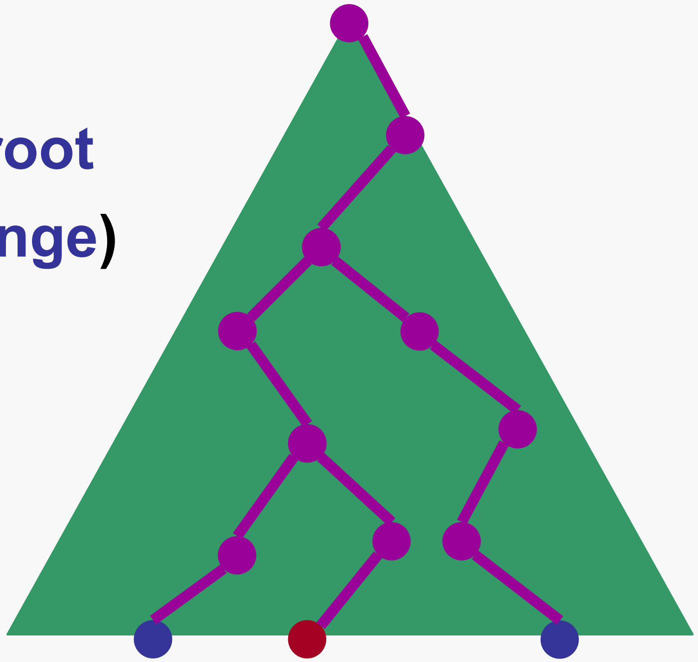


Explore N(ε,δ) independent lassos in the CT

Error margin ε and confidence ratio δ

DRS Las Vegas Approach

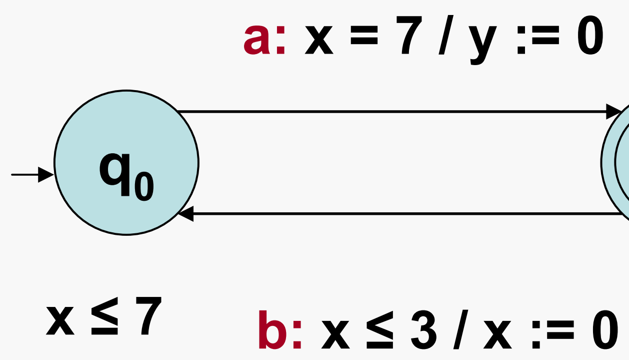
1. take one (several) DRP from the root
2. while (open nodes o are in the fringe)
   1. take a DRP from o
   2. if (accepting) return path
3. return null



A deep random path (DRP) is finished at node o if:

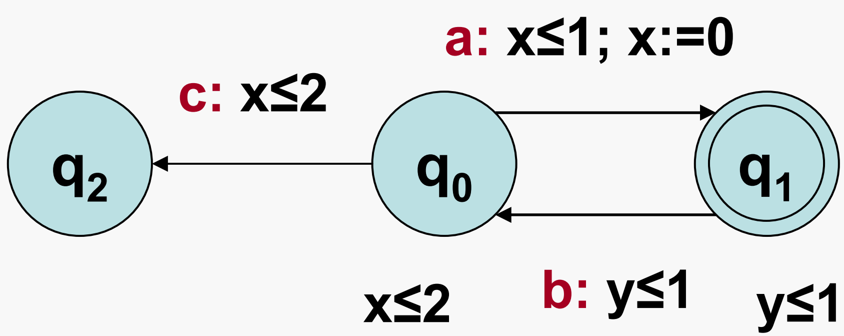
- the maximum depth is reached at o
- no unvisited node is a successor of o

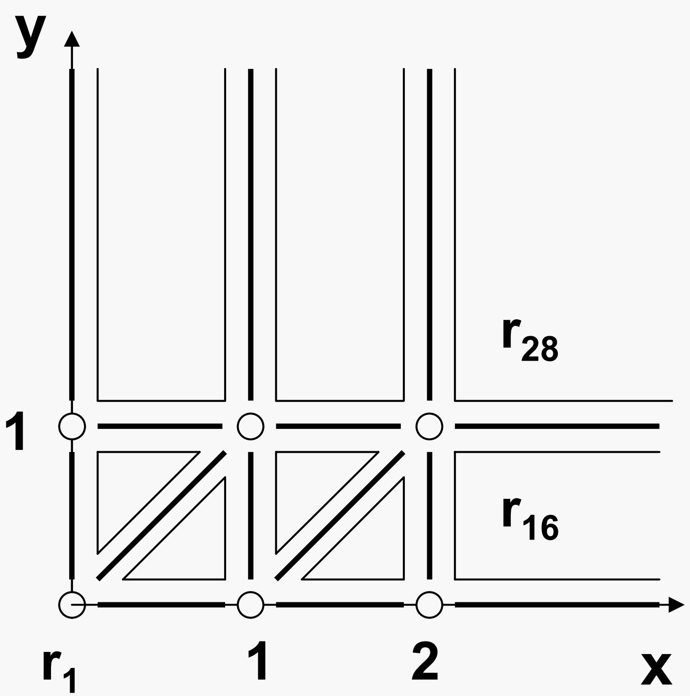
Timed Automata



- Finite set of clocks
- Finite set of discrete states (modes)
- Finite set of accepting states (accepting modes)
- Finite set of edges
  - Guard: convex clock polyhedron
  - Reset: set of clocks to be reset
- State invariant (convex clock polyhedron)

TA and Clock Regions Reduction

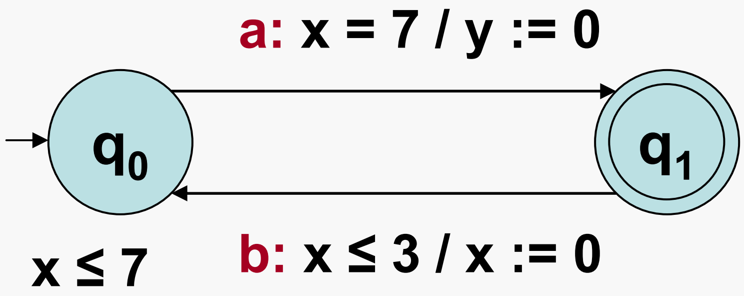


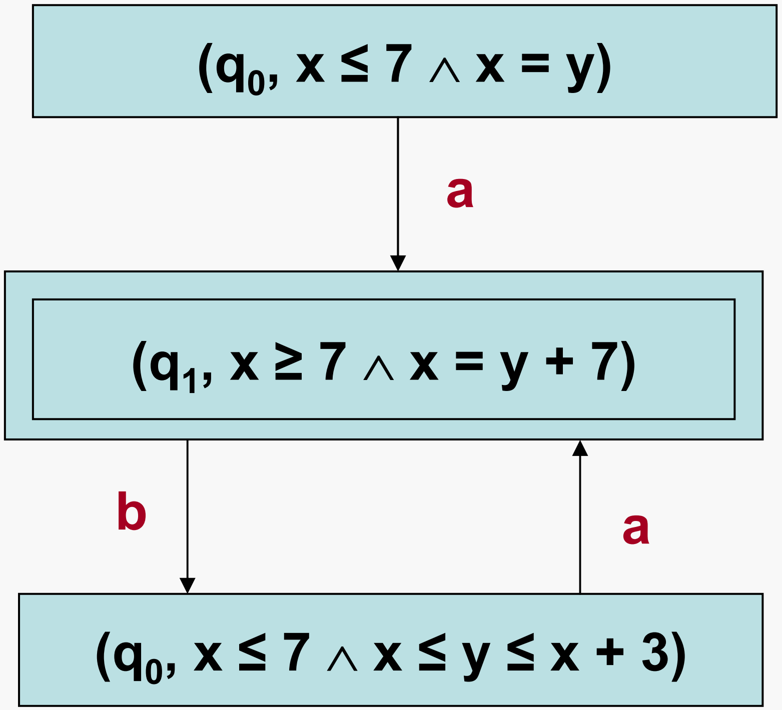


The number of clock regions is exponential in:

- the number of clocks
- the largest clock-upper-bound constant

TA and their Simulation Graph





Experiments

- **DRS implementation:**
  - Extension to Open-Kronos MC for TA
- **Open-Kronos:**
  - Input: a system of TA and a bool exp (accepting states)
  - Translation: to a C-program compiled & linked to Profounder
  - Profounder: on-the-fly gen. of SG and DFS reachability anal.
- **Testbed:**
  - PC equipped with Athlon 2.6GHz
  - 1Gbyte RAM
  - Linux 2.6.5 (Fedora Core)

MutExcl: Buggy Fischer Protocol

	Open Kronos			DRS			UPPAAL
proc	time	states	depth	time	states	depth	time
2	0.04	63	44	0.003	20	6	0.021
4	2.968	1227	1166	0.006	67	28	0.041
8	13.20	35409	2048	0.082	216	211	1.28
12	204	332253	2048	0.512	386	374	18.61
16	>12h	N/A	N/A	0.906	238	222	223 (oom)

MutExcl: Correct Fischer Protocol

	Open Kronos		DRS	UPPAAL
proc	time	states	time	time
2	0.004	203	0.011	0.02
3	0.386	24949	0.513	0.03
4	943	3842501	1388	0.14
5	4h	oom	oom	2.01
6	4h	oom	oom	124
7	4h	oom	oom	>5h

Philips Audio Protocol

	Open Kronos				DRS		UPPAAL
sender	time	states	depth	time	states	depth	time
1	0.004	72	71	0.003	16	12	0.026
4	3.259	46263	2048	0.007	30	26	0.041
8	422.2	1026446	2048	0.041	93	26	0.158
12	>12h	N/A	N/A	0.736	375	42	0.802
24	>12h	N/A	N/A	0.02	41	17	39.095
28	>12h	N/A	N/A	0.033	50	22	107 (oom)

B&O Audio/Video Protocol

	Open Kronos			DRS			UPPAAL
sender	time	states	depth	time	states	depth	time
2	0.226	1285	1284	0.034	1659	1657	0.174
3	35.61	1135817	1997	10.76	166113	2318	1.05
4	53.532	1130669	1608	50.554	617760	2972	10.1
5	1200	oom	N/A	10m	6769520	4734	2m
6	1200	oom	N/A	37m	30316978	13376	12m (oom)

Related Work

- Random walk testing:
  - Heimdahl et al: Lurch debugger.
  - P. Haslum: Monte Carlo MC by random walk.
- Random walks to sample system state space:
  - Mihail & Papadimitriou (and others)
- Monte Carlo Model Checking of Markov Chains:
  - Herault et al: LTL-RP, bonded MC, zero/one ET
  - Younes et al: Time-Bounded CSL, sequential analysis
  - Sen et al: Time-Bounded CSL, zero/one ET
- Probabilistic Model Checking of Markov Chains:
  - ETMCC, PRISM, PIOAtool, and others.
- Sat Solvers: randomization is not the main concern

Conclusions

- DRS is a complete randomized, MC algorithm for the classical problem of model checking safety properties.
- DRS allows to fine tune the initial number of random paths from the root and the maximum search depth.
- DRS is able to find extremely deep counterexamples while consistently outperforming Kronos and UPPAAL in the process.
- DRS is best suited to find counterexamples and not to prove their absence.