Game-based and Simulation-based Improvements for LTL Synthesis

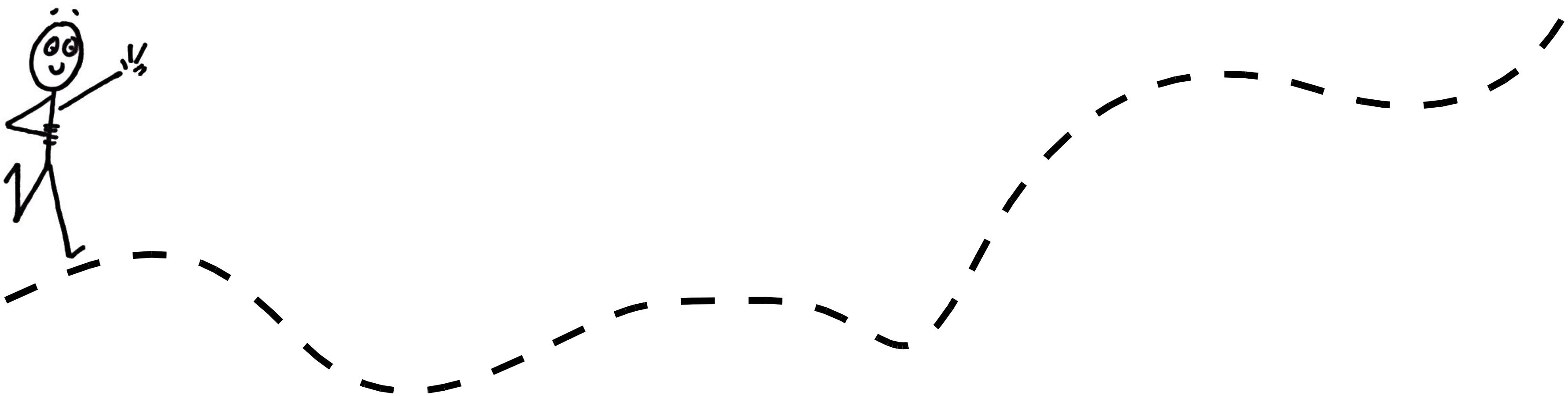
Barbara Jobstmann
Roderick Bloem

Graz University of Technology, Austria

August 21st, 2006

Motivation

- Synthesis from specification fascinating
- Correct by construction - no verification
- You say what, it tells how!
- Goes back to Church (1962)
- What has changed since then?



Outline

- Introduction
- Safraless Approach
  - Steps and Improvements
- Lily
- Conclusion

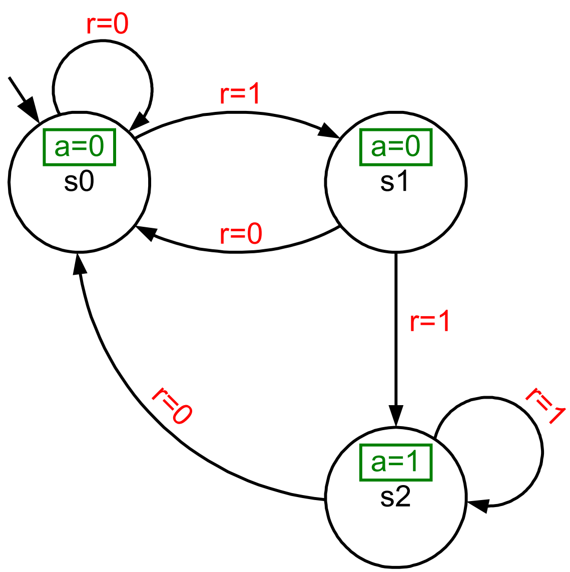
LTL Synthesis

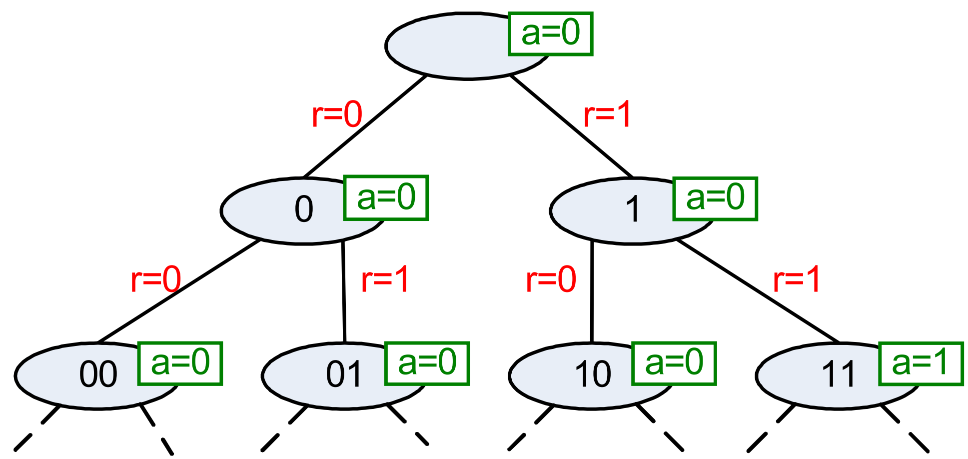
- Automatically build design from properties
- Input
  - Set of LTL properties
  - Partition of the atomic propositions (input/output signals)
- Output
  - Automatically created functionally correct finite-state machine (Moore)

- Proposed for LTL by Pnueli, Rosner (POPL'89)

FSM and Trees

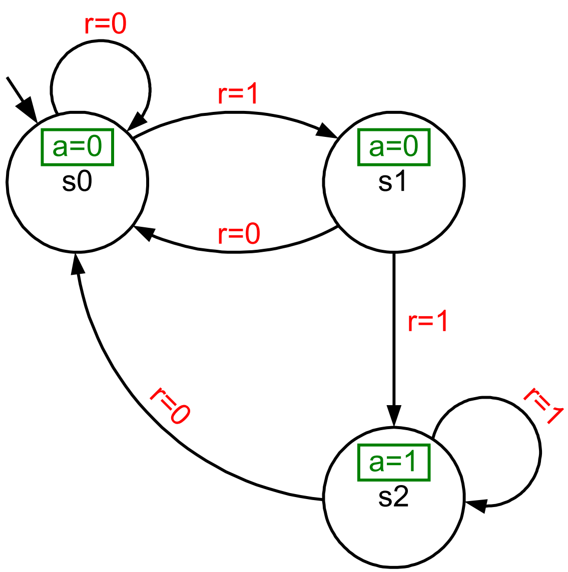
- Moore machine
  - r=1,r=0 ....input alphabet
  - a=1,a=0 .. output alphabet
  - Input signal i, output signal o
- Tree (regular)
  - r=1,r=0 ....directions D
  - a=1,a=0 .. alphabet Σ (labeling)

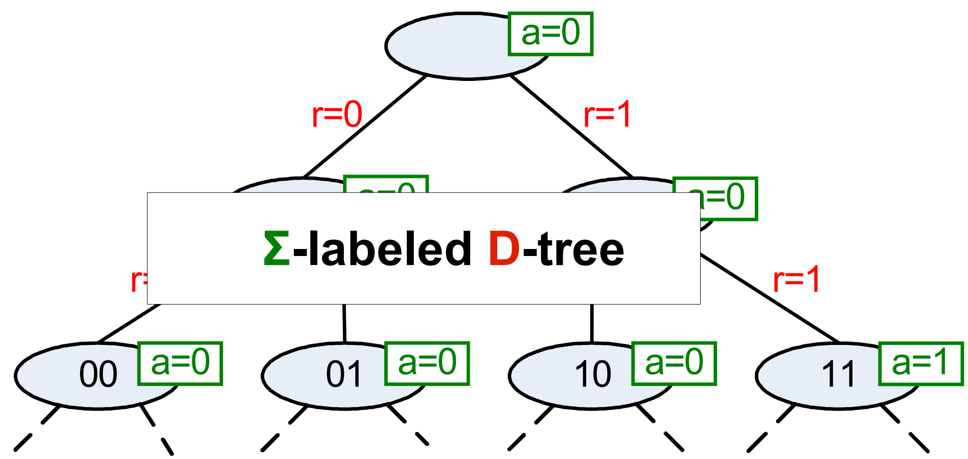




FSM and Trees

- Moore machine
  - r=1,r=0 ....input alphabet
  - a=1,a=0 .. output alphabet
  - Input signal i, output signal o
- Tree (regular)
  - r=1,r=0 ....directions D
  - a=1,a=0 .. alphabet Σ (labeling)





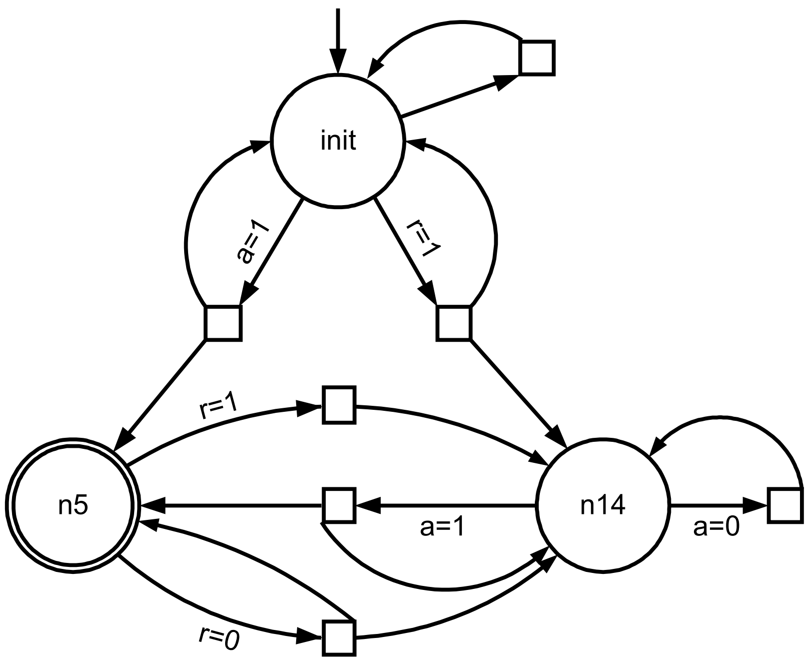
Idea

1) Build a tree automaton
   - Directions are input values ($D=2^I$, input signals I)
   - Alphabet are output values ($\Sigma=2^O$, output signals O)
   - Automaton accepts all **Σ-labeled D-trees** where all paths satisfy the given formula
2) Compute language emptiness
3) Build FSM from the witness (a Σ-labeled D-tree)

Necessary Theory

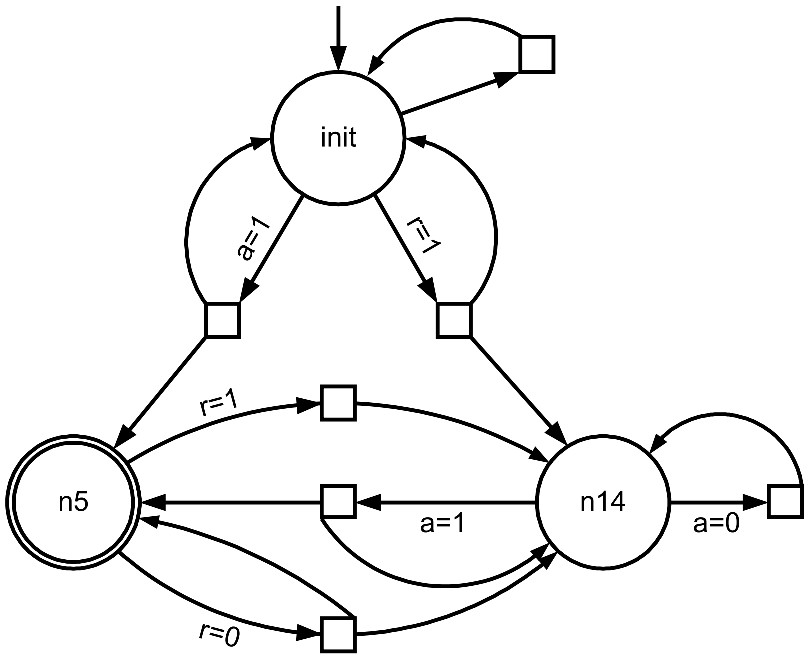
- Infinite game theory
- Automata theory
  - Branching mode (**N**ondeterministic, **U**niversal, **A**lternating)
  - Acceptance condition (**B**üchi, **C**o-Büchi, **W**eak, ..)
  - Input element (**W**ord,**T**ree)
  - Use of KV's abbreviation (e.g.,NBW,UCT,...)

Alternating Word Automata

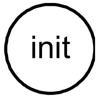


- N+U branching (edges we can follow and edges we must follow)
- Notation:
  - Circles represent states
  - Boxes represent universal edges
  - Edges are labeled with sets of labels

ABW

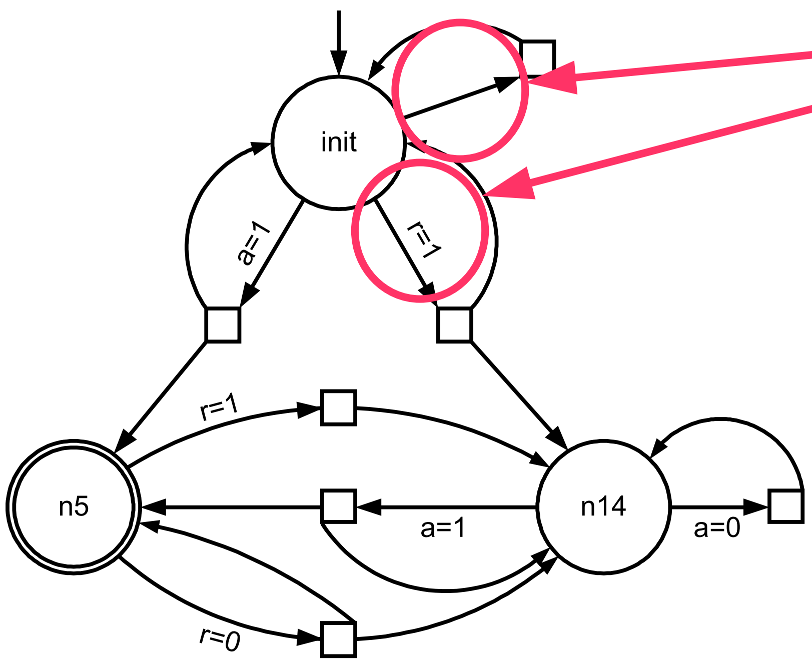


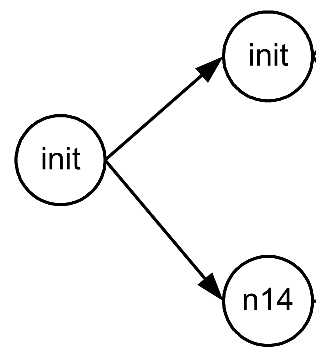
Input word:	r=1 a=0	r=1 a=0	r=0 a=1	...



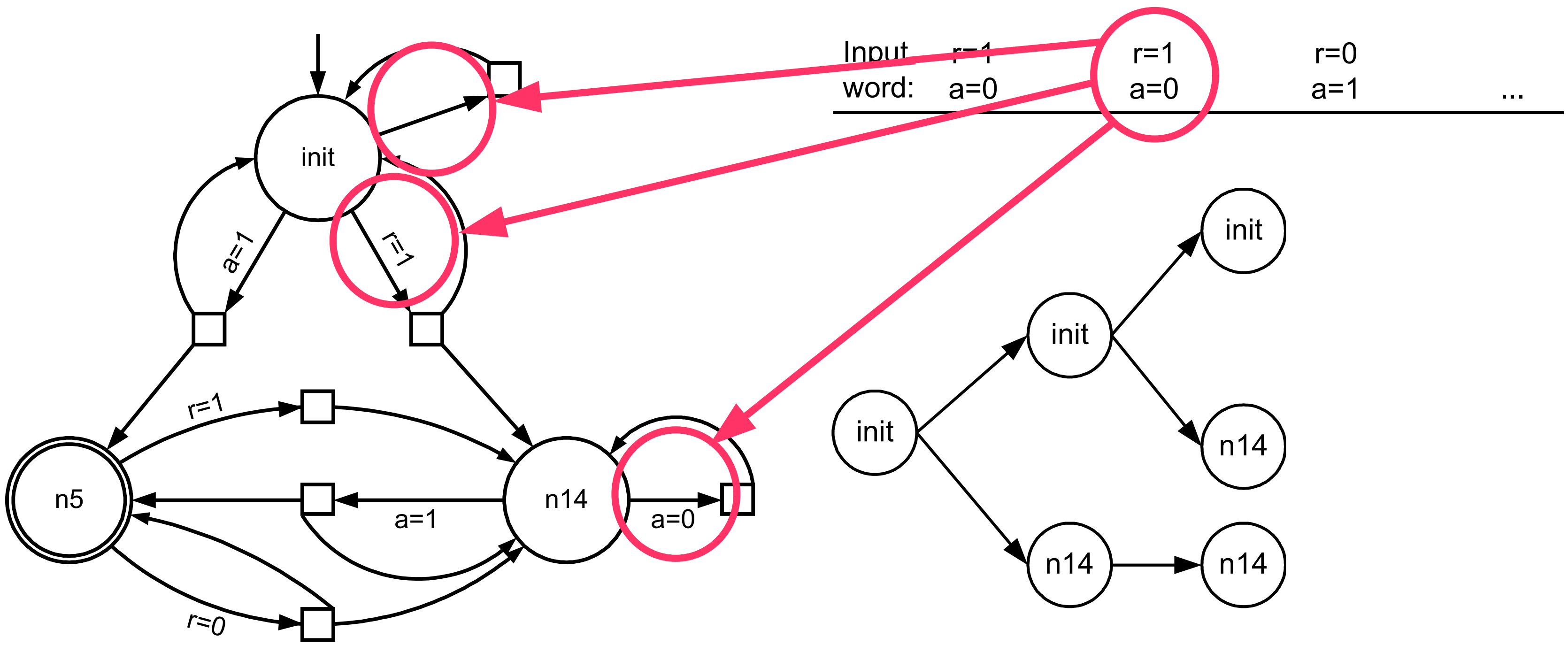
ABW

Input word:	r=1 a=0	r=1 a=0	r=0 a=1	...

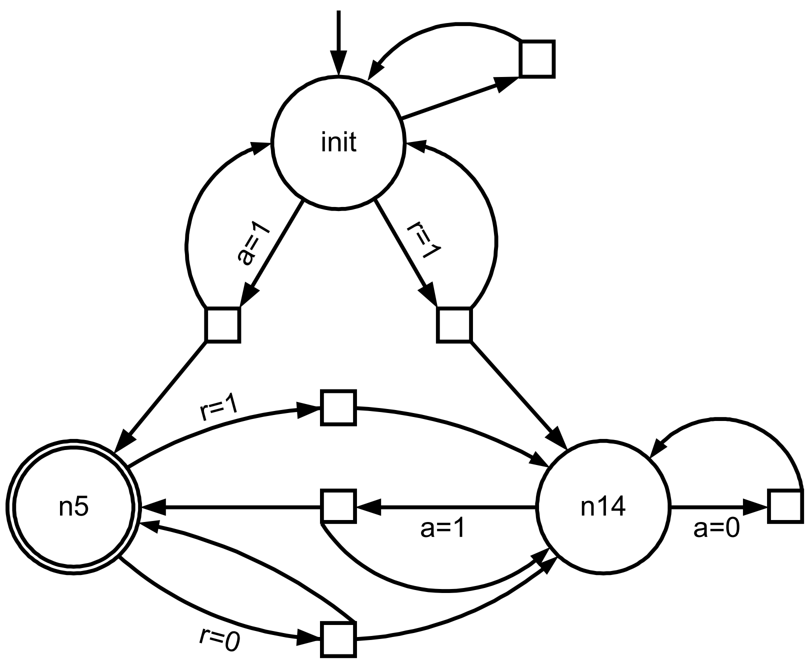


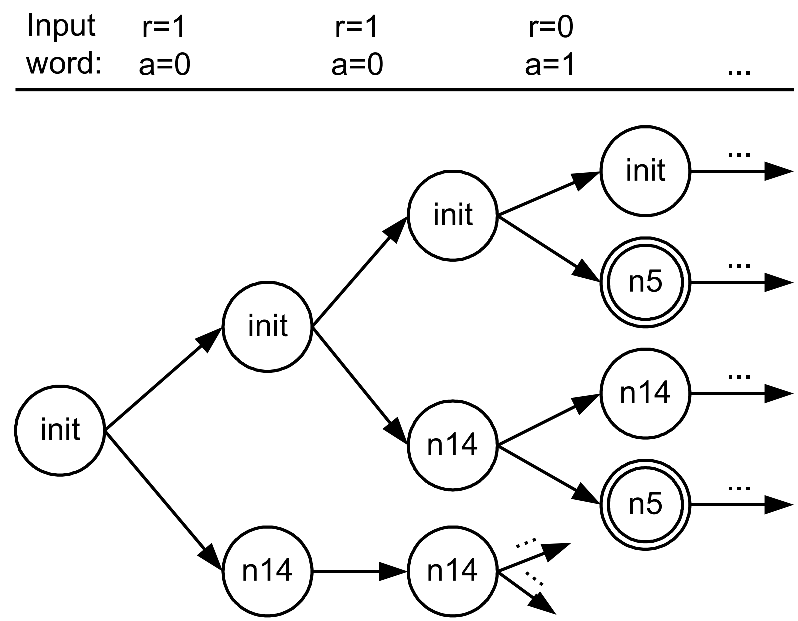


ABW

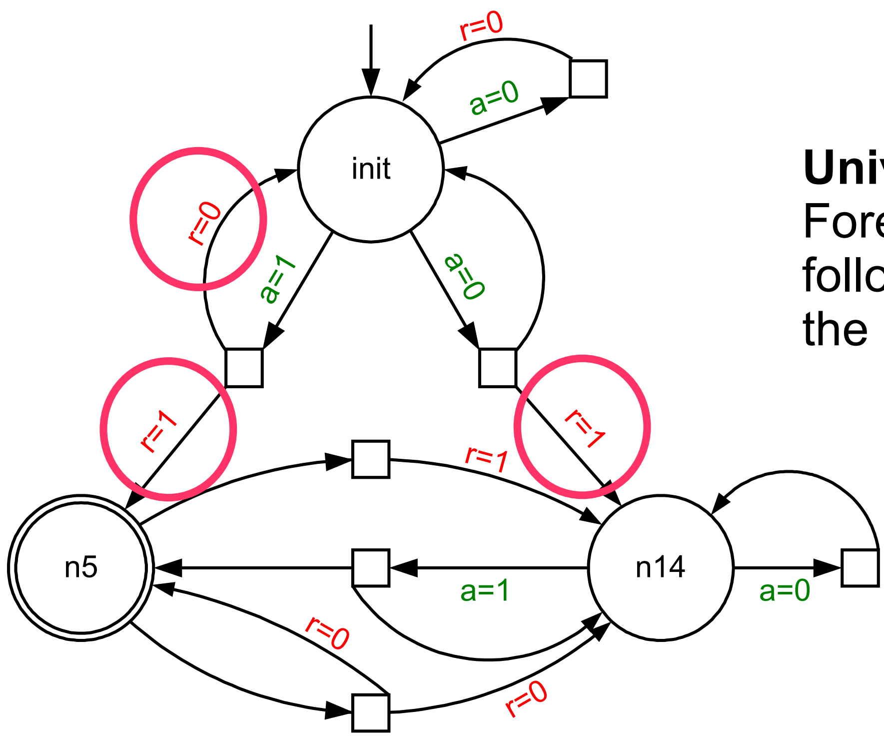

Input word: r=1 a=0 | r=1 a=0 | r=0 a=1 | ...
init
a=1
r=1
n5
n14
r=1
a=1
a=0
r=0
init
init
init
n14
n14
n14

ABW

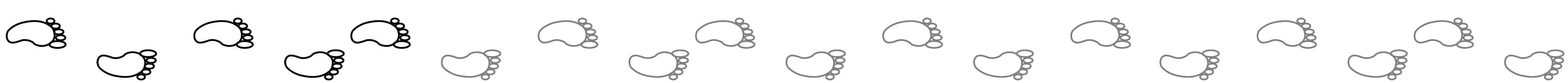

init
a=1
r=1
r=1
n5
n14
a=1
a=0
r=0

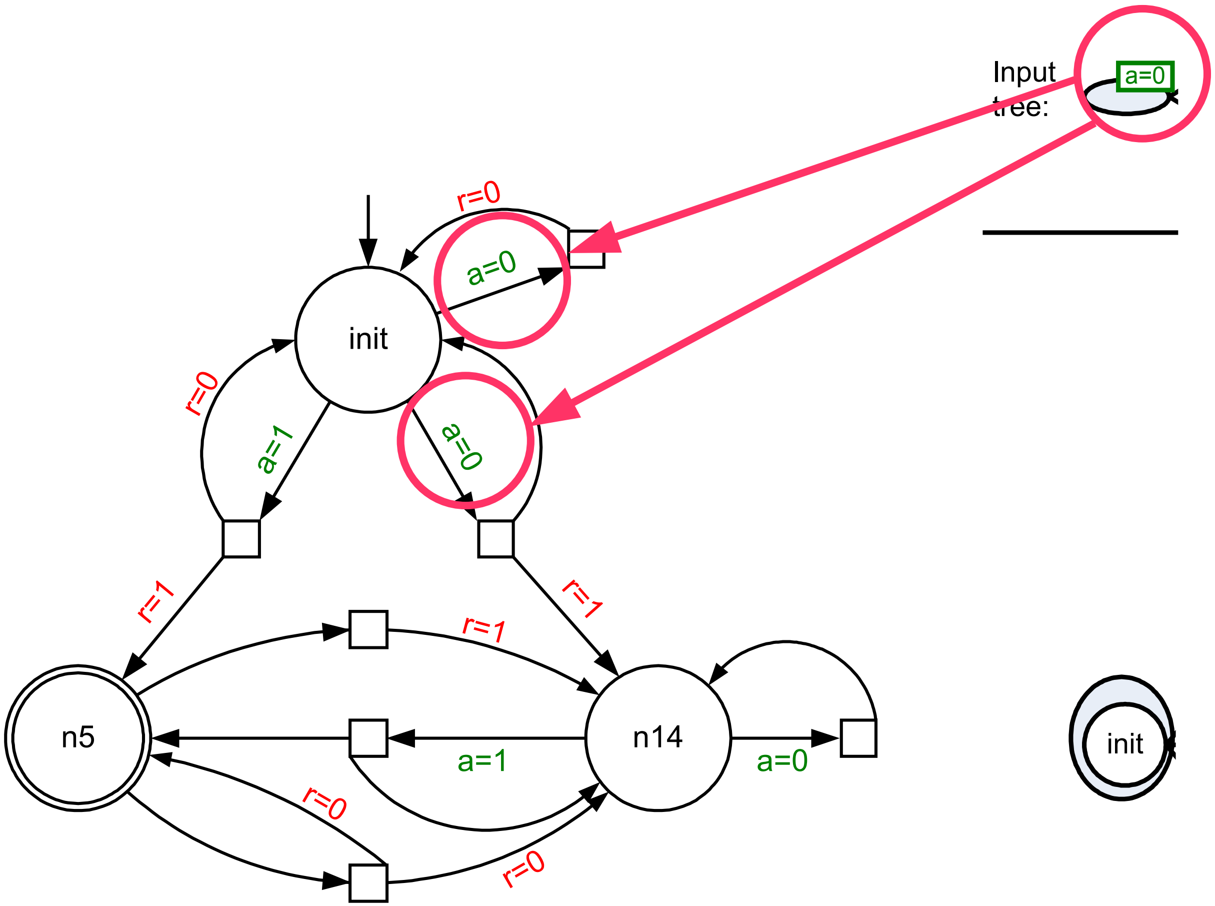

Input word:
r=1 a=0
r=1 a=0
r=0 a=1
...
init
init
init
init
n5
n14
n14
n5
n14
n14
...

Tree Automata

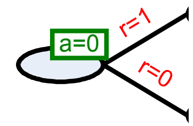


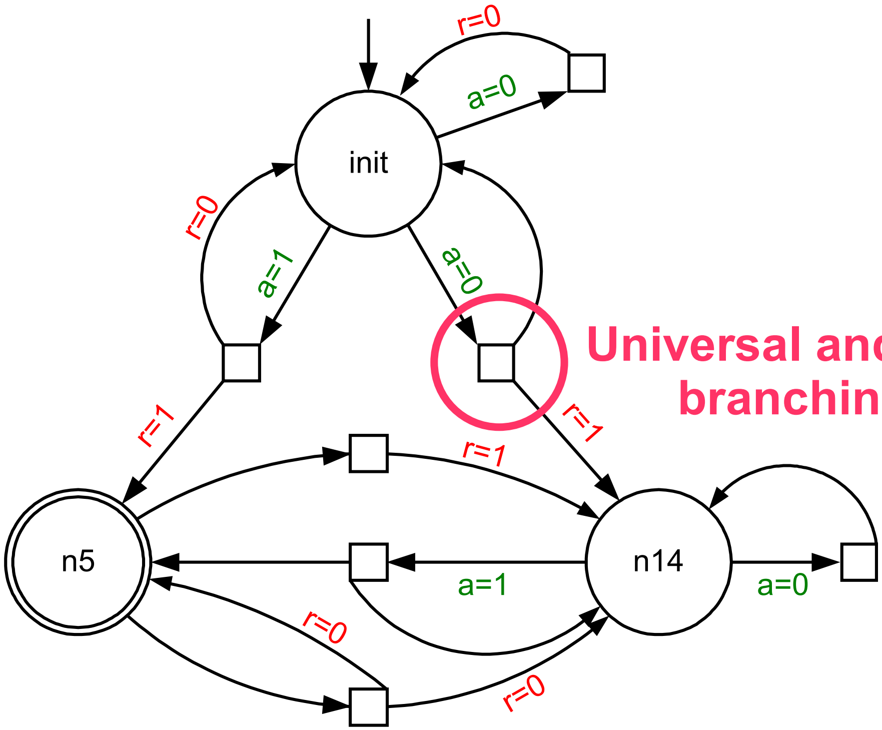
Universal edges:
Foreach direction,
follow only
the matching edges



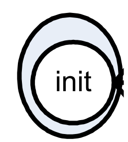

Input
tree:
a=0
r=0
a=0
init
r=0
a=1
a=0
r=1
r=1
r=1
n5
a=1
n14
a=0
r=0
r=0
init

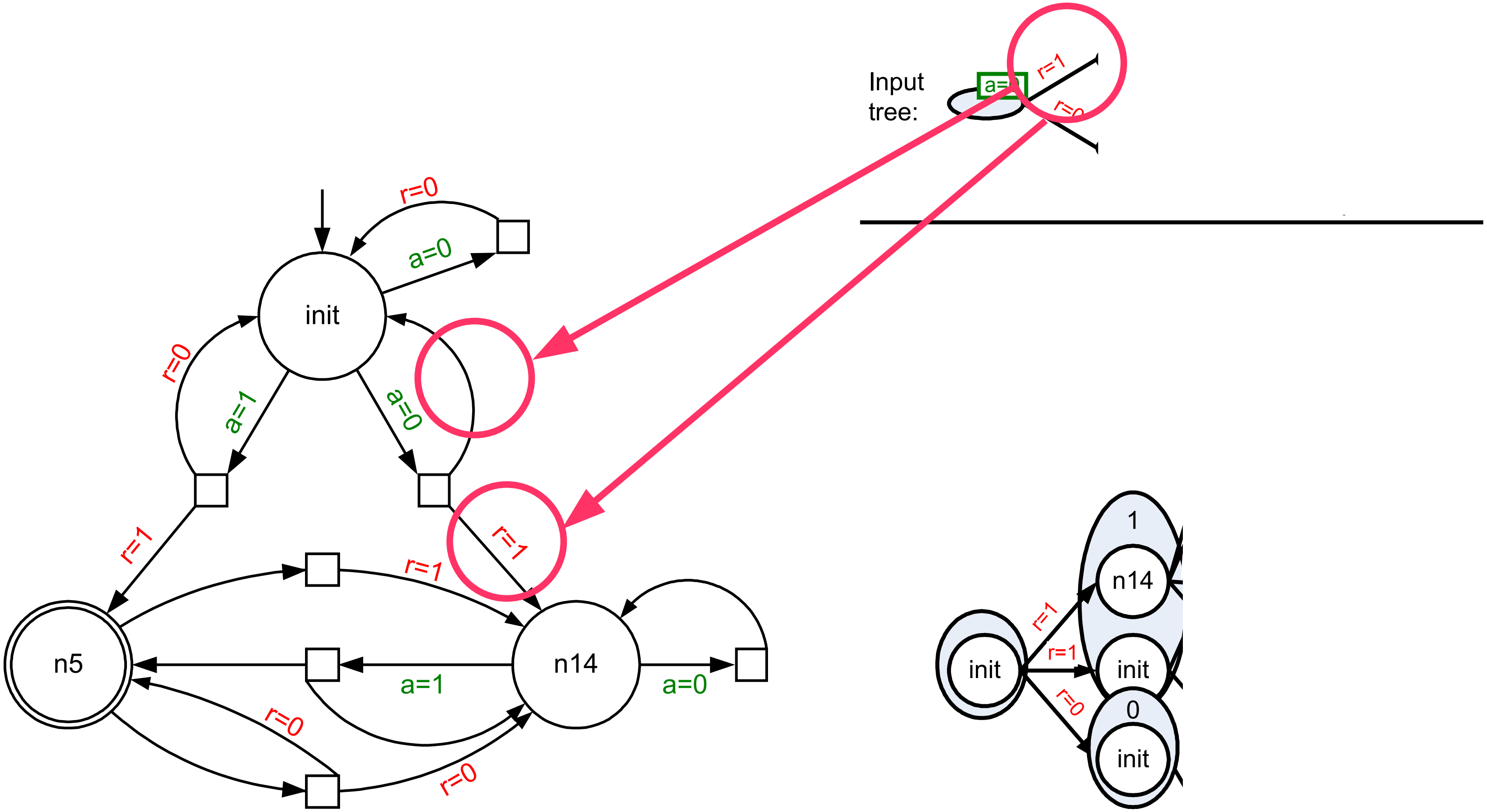
Input tree:

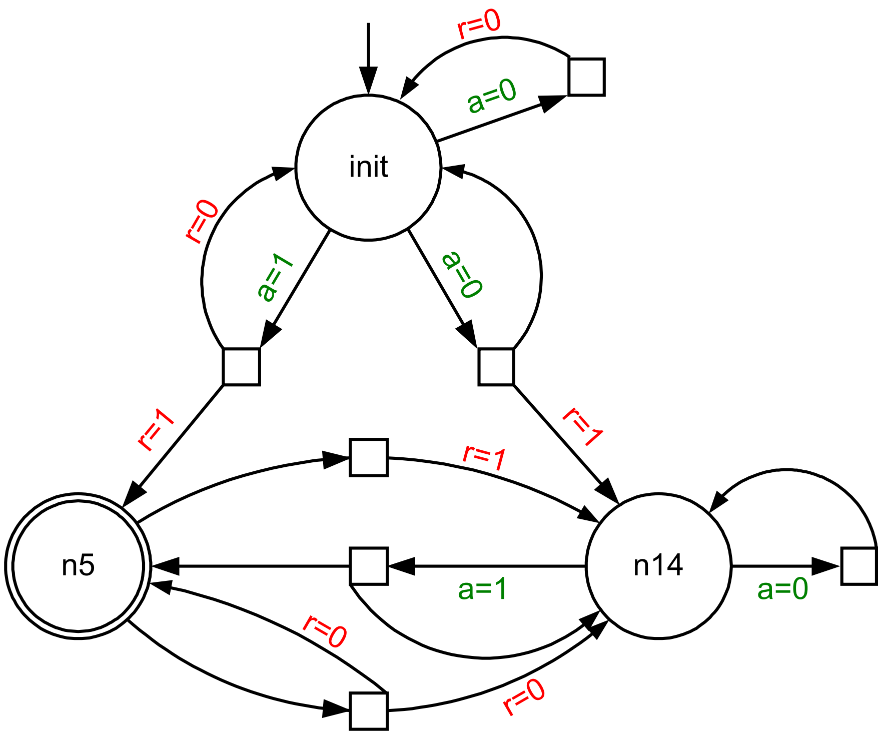




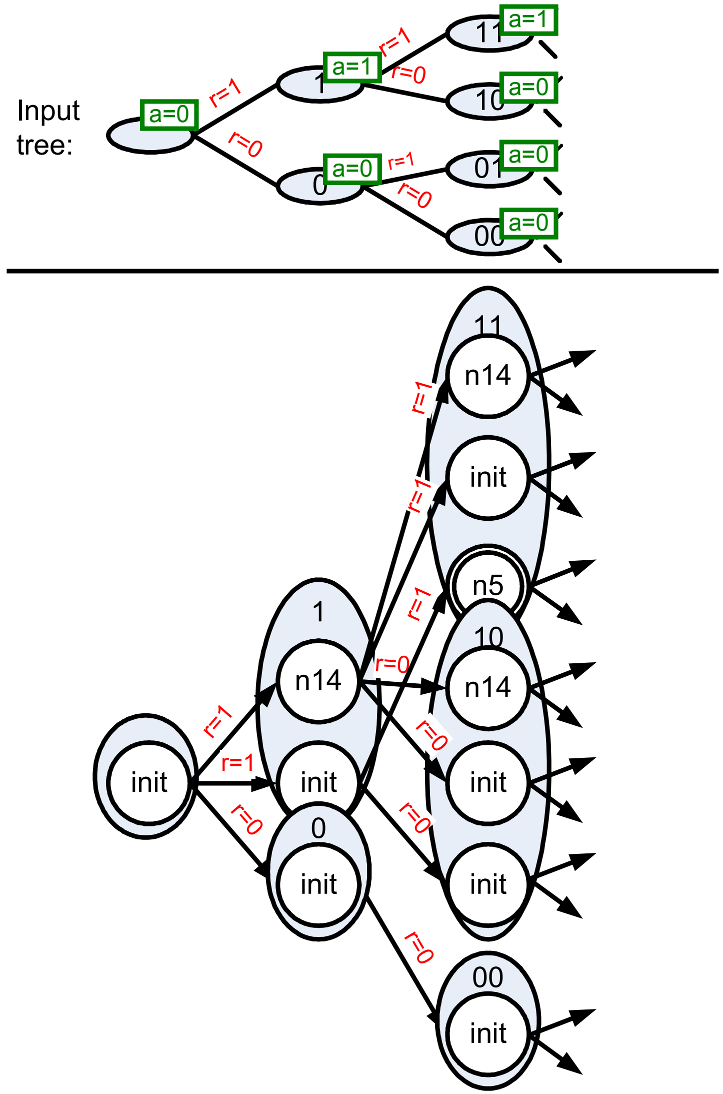
Universal and tree branching




Input
tree:
a=1
r=1
r=0
init
n14
n5
a=0
a=1
r=0
r=1
1
0



init
n5
n14
r=0
a=0
r=0
a=1
a=0
r=1
r=1
r=1
a=1
a=0
r=0
r=0



Input tree:
a=0
r=1
r=0
1
0
a=1
a=0
r=1
r=0
11
10
01
00
a=1
a=0
a=0
a=0
init
1
0
11
10
00
n14
n5
init
r=1
r=0

Standard Approach [PR89]

- Construct a nondeterministic Büchi automaton
- Determinize it with Safra's determinization construction
  → Rabin word automaton
- Convert to Rabin tree automaton
- Check Language Emptiness

Issues

- 2EXP worst case complexity
- Safra's determinization construction

Solutions

- Concentrate on subsets
  - Alur, Madhusudan, Nam (BMC'03, STTT'05)
  - Wallmeier, Hütter, Thomas (CIAA'03)
  - Harding, Ryan, Schobbens (TACAS'05)
  - Piterman, Pnueli, Sa'ar (VMCAI'06)
- Full LTL (Safraless approach)
  - Kupferman, Vardi (FOCS'05)
  - Kupferman, Piterman, Vardi (CAV'06)

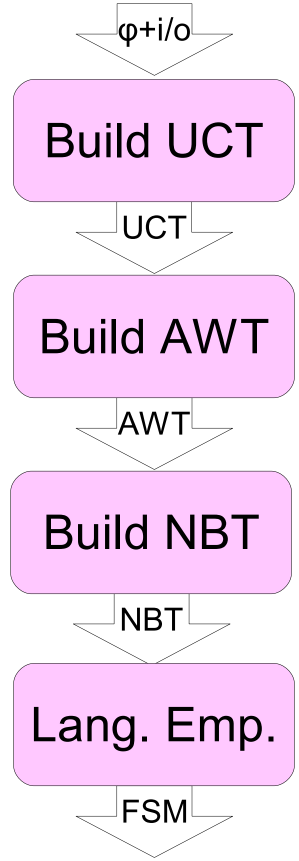
Safraless Approach [KV05]

1) Build a UCT
   - Negate φ
   - Build an NBW for ¬φ
   - Invert NBW → UCT

2) Convert to AWT

3) Convert to NBT

4) Check Language Emptiness



$L(UCT) = L^T(\varphi)$

$L(AWT) \subseteq L^T(\varphi)$

$L^T(\varphi) \neq \emptyset \rightarrow L(AWT) \neq \emptyset$

$L(NBT) \subseteq L^T(\varphi)$

$L^T(\varphi) \neq \emptyset \rightarrow L(NBT) \neq \emptyset$

Safraless Approach

φ+i/o

Build UCT

UCT

Build AWT

AWT

Build NBT

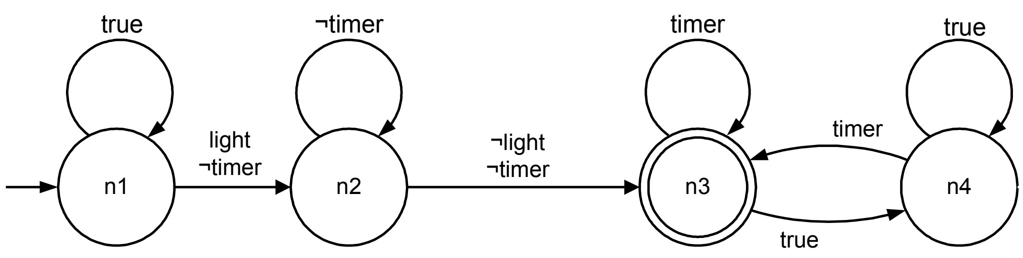
NBT

Lang. Emp.

FSM

Build a UCT for φ (1)

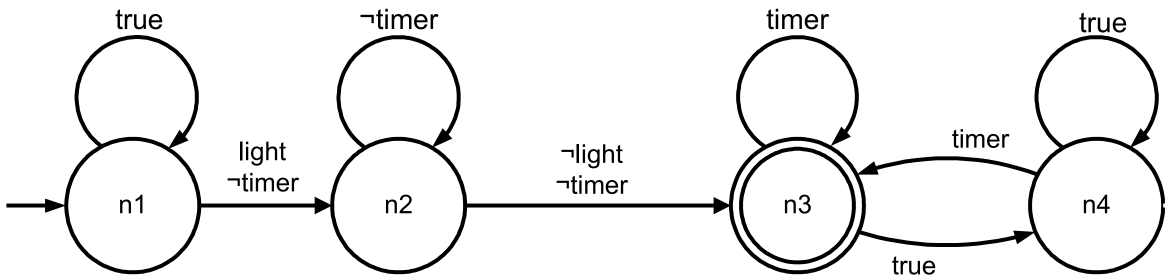
- NBW for ¬φ
  - Build a generalized NBW
  - Apply Counting construction → Single fairness condition
  - Optimizations of Somenzi, Bloem (CAV'00)
- Example:
  - φ=G(F(timer)) →G(light →light U timer))
  - NBW for ¬φ build with Wring

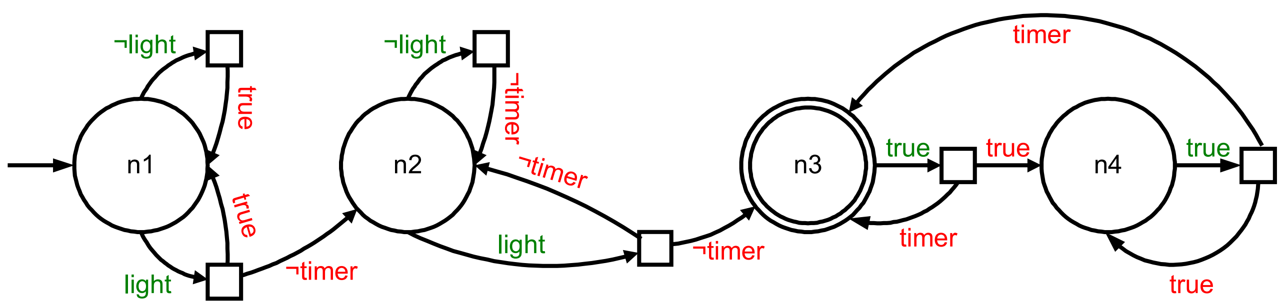


Build a UCT for φ (2)

- Dualize:
  - Nondeterminism → Universality
  - Büchi → Co-Büchi (accepting states → α-states)
- Word → Tree
  - Inputs → Directions (Universality)
  - Outputs → Labels
- Very similar structure

UCT Example





$$\varphi = G(F(timer)) \rightarrow G(light \rightarrow light\ U\ timer))$$

UCT
AWT
NBT
LE

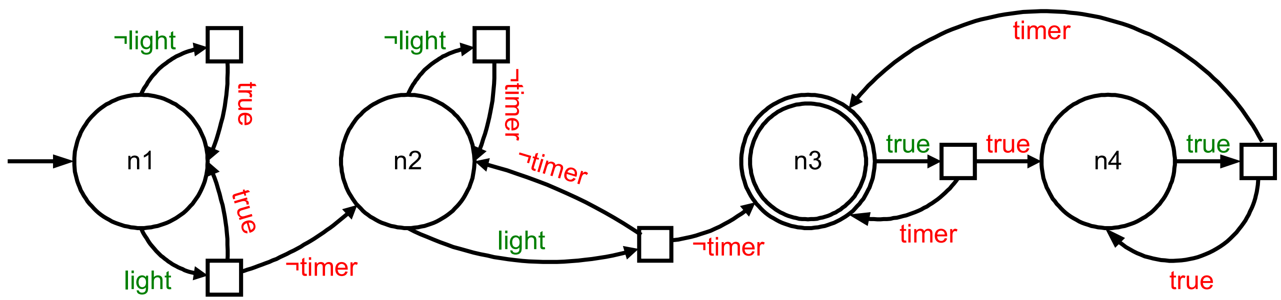
Simplify UCT

- Idea
  - Approximate states with empty language (accept no tree)
  - Trees with non-accepting path are rejected
  - Environment can force a non-accepting path
  - Sufficient (but not necessary) for language emptiness

UCT
AWT
NBT
LE

Simplify UCT

- Game
  - System picks the label and the nondeterminism
  - Environment picks direction and universality
- State s is winning for environment → $L^T(s)$ empty

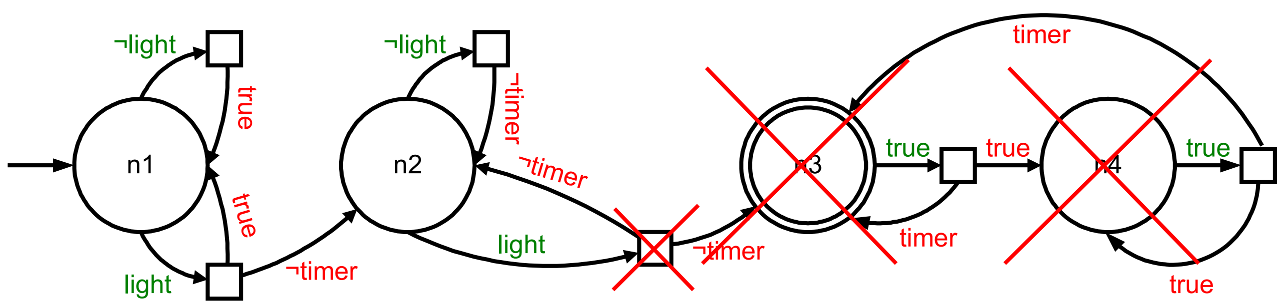


n3...α-state (co-büchi)

UCT
AWT
NBT
LE

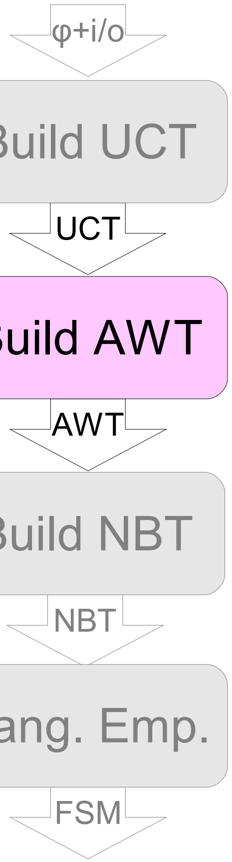
Simplify UCT

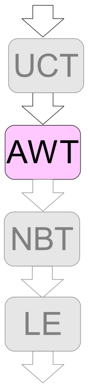
- Game
  - System picks the label and the nondeterminism
  - Environment picks direction and universality
- State s is winning for environment → $L^T(s)$ empty



n3...α-state (co-büchi)

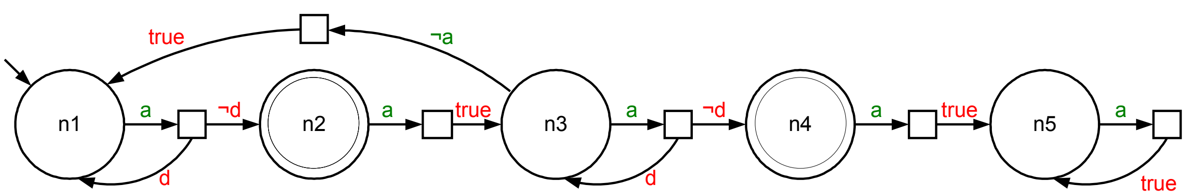
Safraless Approach





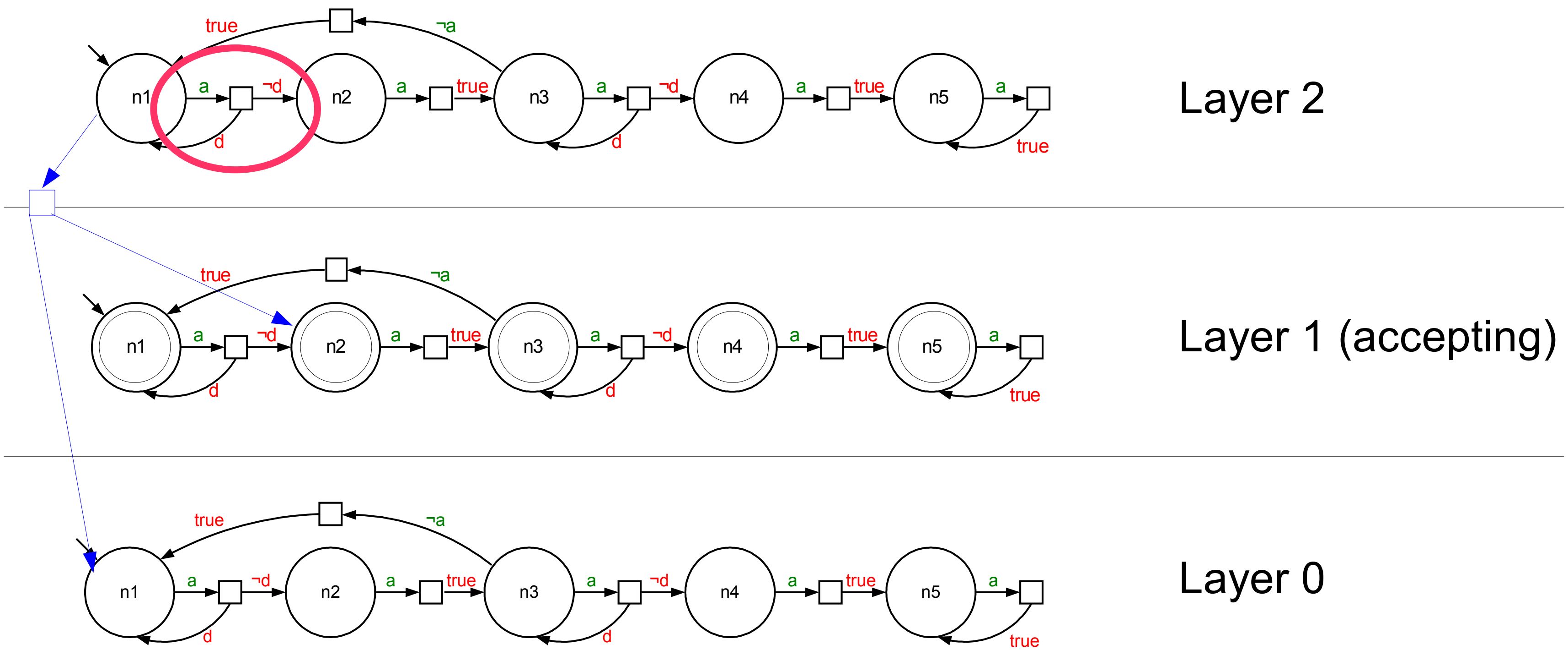
Build AWT

- Idea
  - UCT accepts if α-states avoided
  - Build layers by copying the UCT
  - Alternate accepting and non-accepting layers
  - Tr of α-states in accepting layers is false
  - In accepting layers visit to α-states not possible → only finitely many visits to α-states
- Example



n2,n4...α-states (co-büchi)

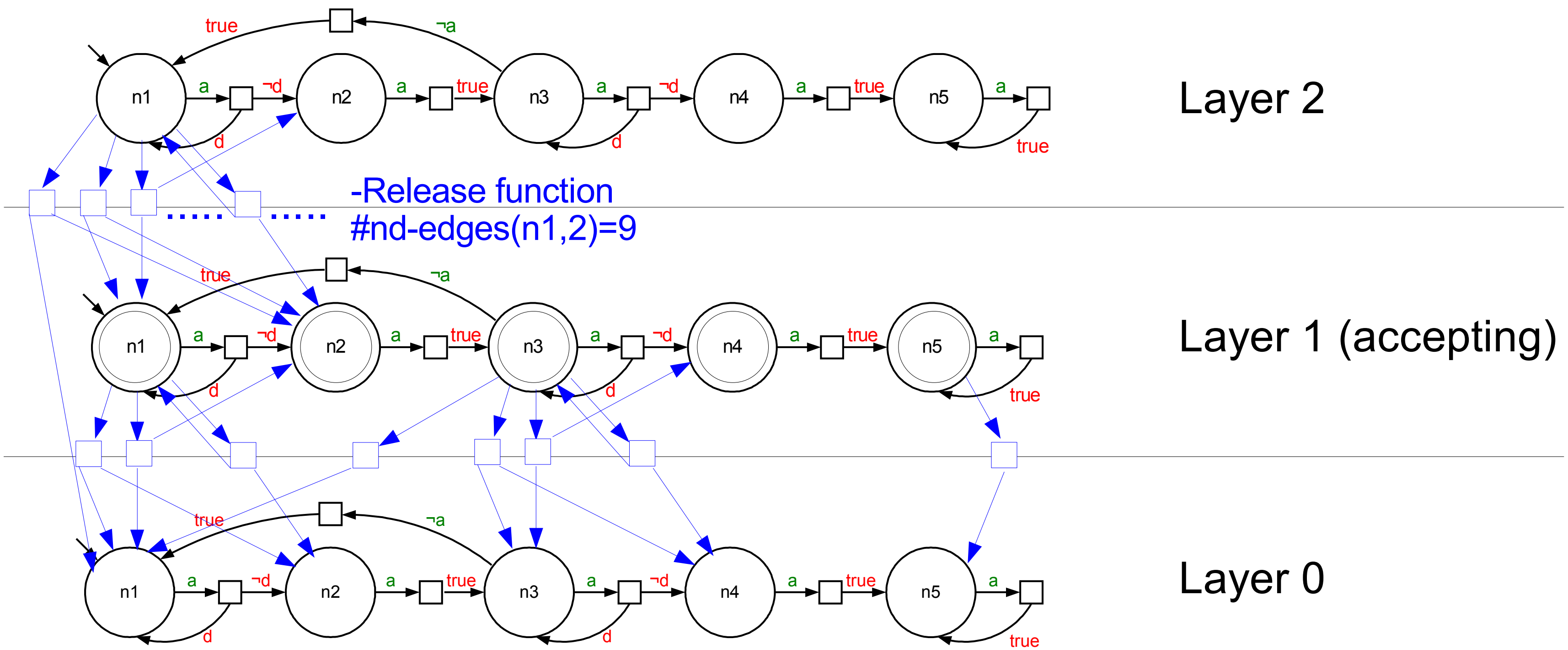
AWT Example



n2,n4…α-states (co-büchi)

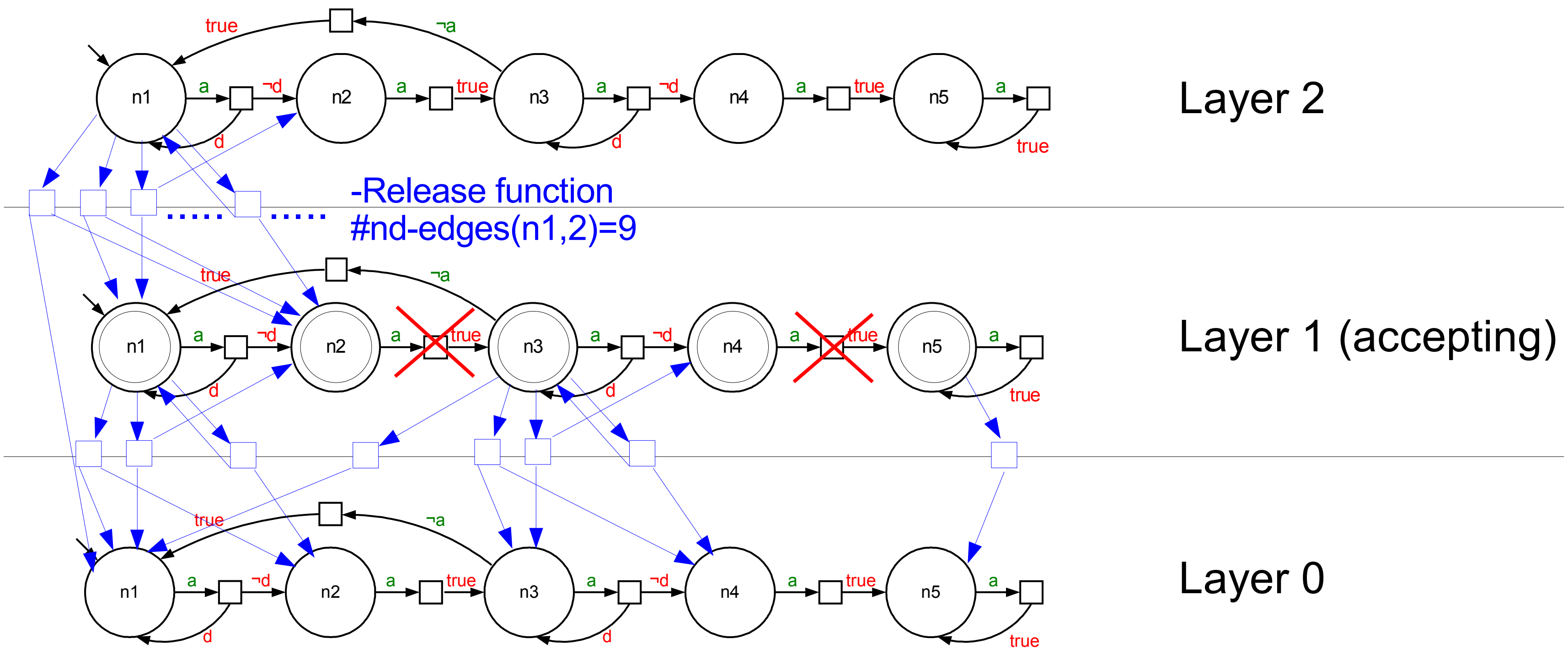
UCT
AWT
NBT
LE

AWT Example



n2,n4...α-states (co-büchi)

AWT Example



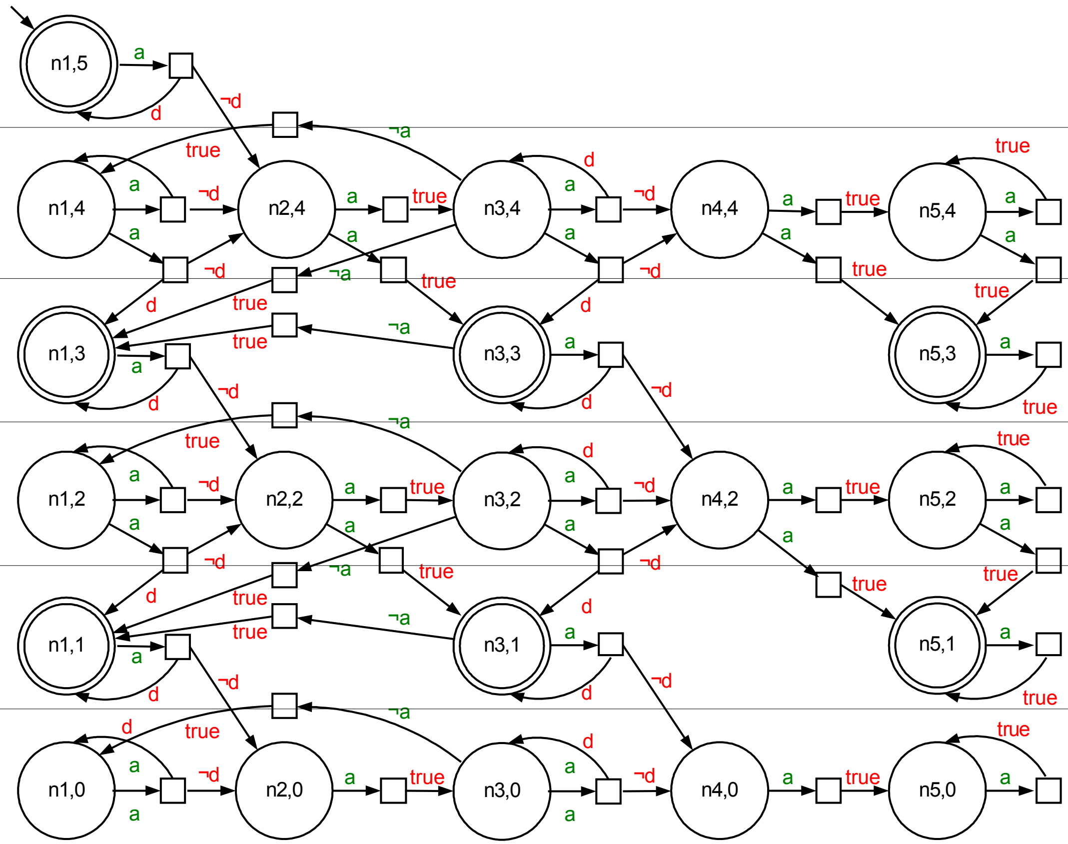
n2,n4...α-states (co-büchi)

Simplify AWT: Construction

- Simplify release function
  - cf. Gurumurthy, Kupferman, Somenzi, Vardi (CHARME'03)
- Ideas
  - Release just between two layers (from i to i and i-1)
  - Stay in accepting layer if possible
  - Do not distinguish between directions if same state

UCT
AWT
NBT
LE

Simpler Release Function



Layer 5 (accepting)

Layer 4

Layer 3 (accepting)

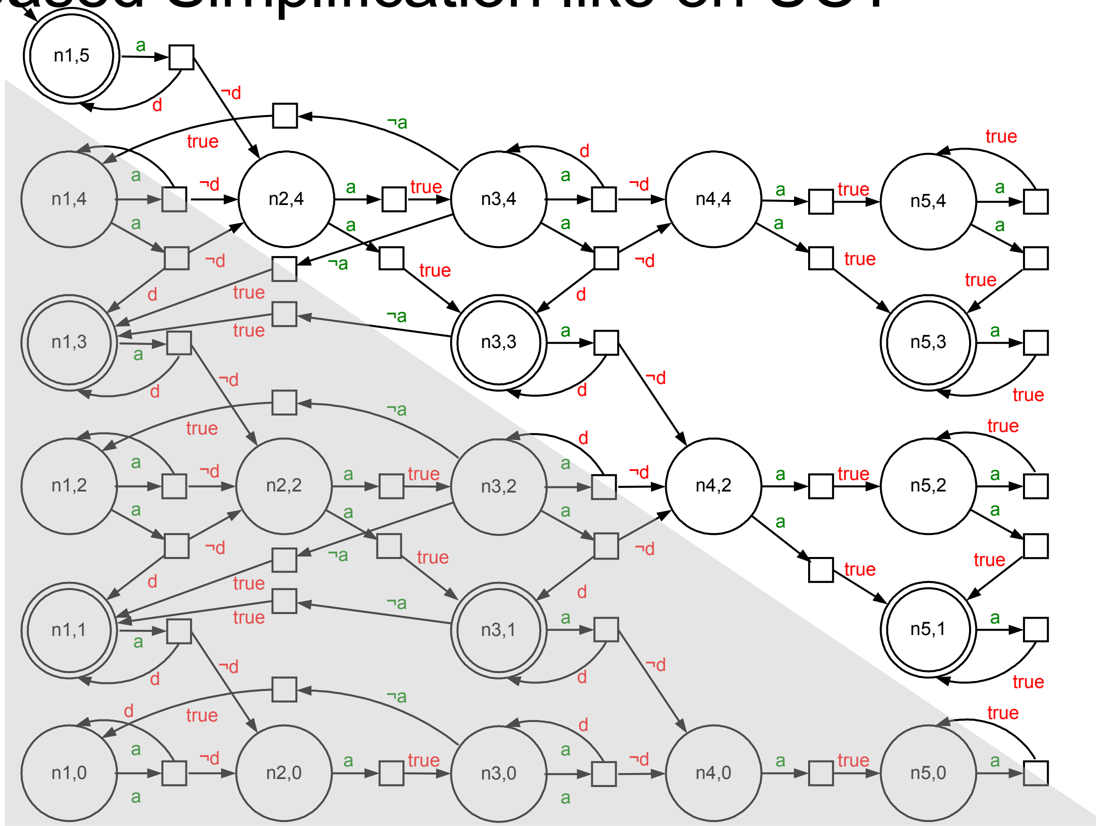
Layer 2
#nd-edges(n1,2)=2

Layer 1 (accepting)
#edges < #edges of UCT

Layer 0

Simplify AWT: Game-based

- Game-based Simplification like on UCT



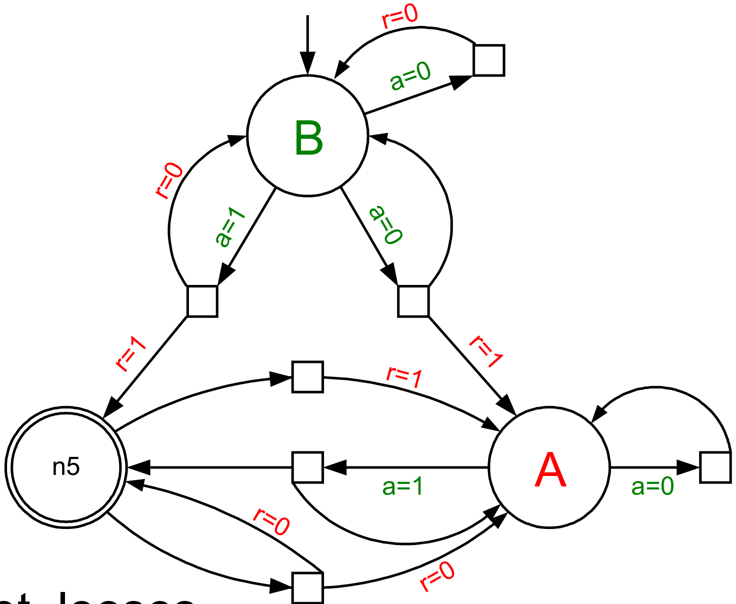
Simplify AWT: Simulation-based

- Simulation relation for tree automaton
  - cf. Alur, Henzinger, Kupferman, Vardi (CONCUR'98)
  - cf. Fritz, Wilke (FSTTCS'02)
- Idea:
  - State B simulates state A $\rightarrow$ tree accepted by A is accepted by B, $L^T(A) \subseteq L^T(B)$
  - Nondeterministic edges to A and B, remove edge to A
  - Universal edges to A and B, remove edge to B
- Use a game to compute simulation relation

UCT
AWT
NBT
LE

Simplify AWT: Simulation-based

- Turn based game between simulated and simulating state
- Game for B simulates A
  - A picks label and his nondeterminism
  - B picks her nondeterminism
  - A picks direction
  - A picks universality of B
  - B picks universality of A
- Winning Condition
  - Player who has to choose from empty set, looses
  - B wins a play, if whenever A is in an accepting state also B is in an accepting state (direct simulation)



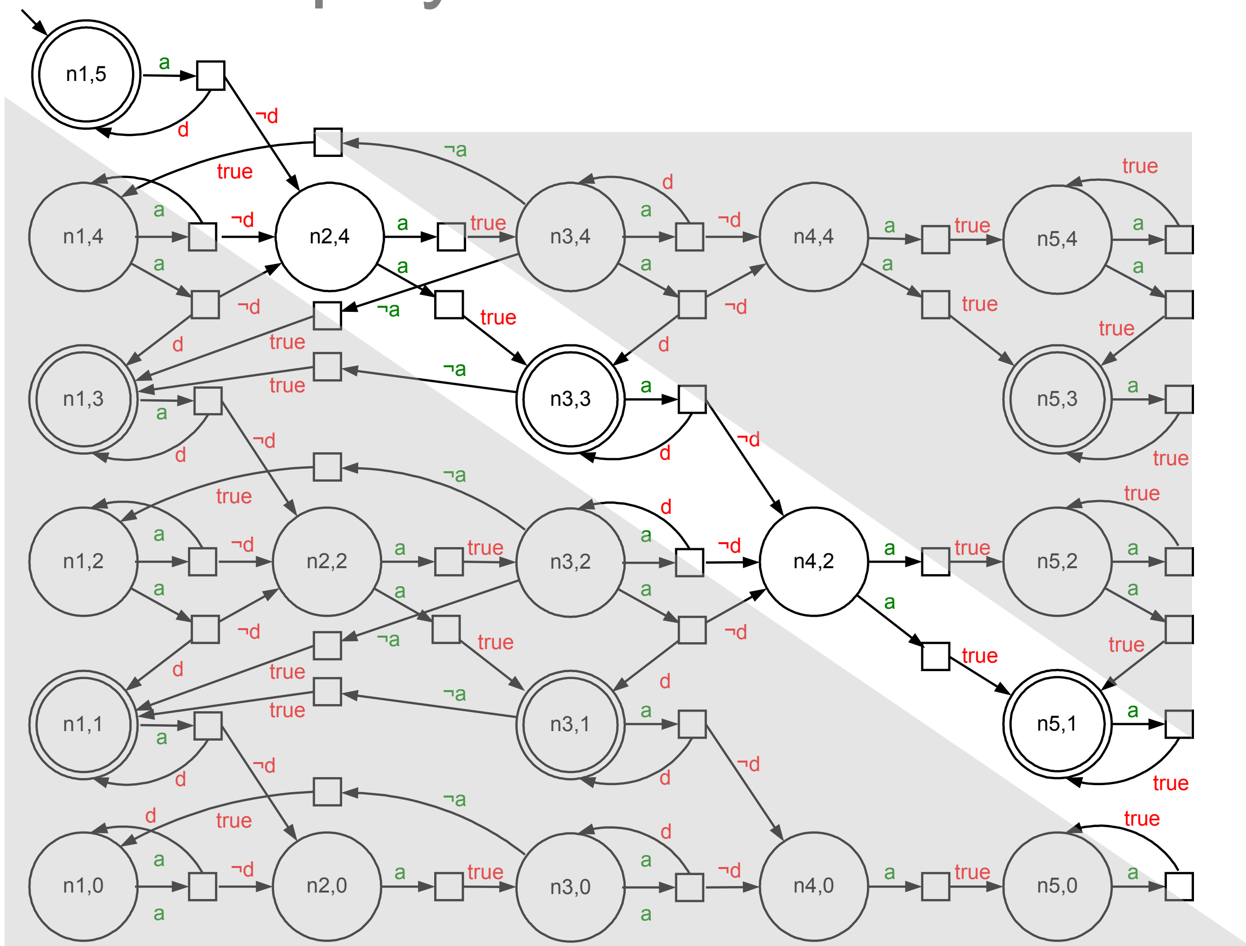
UCT

AWT

NBT

LE

Simplify AWT: Simulation-based



Sim. equivalent

$n4,4 =_{AWT} n4,2$

$n5,4 =_{AWT} n5,2$

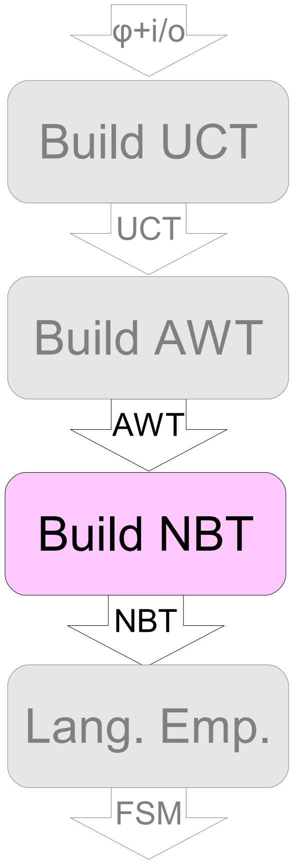
$n5,3 =_{AWT} n5,1$

A simulated by B

$n3,4 \leq_{AWT} n3,3$

$n5,2 \leq_{AWT} n5,1$

Safraless Approach



Build NBT

- Tree-version of Miyano-Hayashi's construction
- Extended subset construction
- NBT-state ≈ set of AWT-states

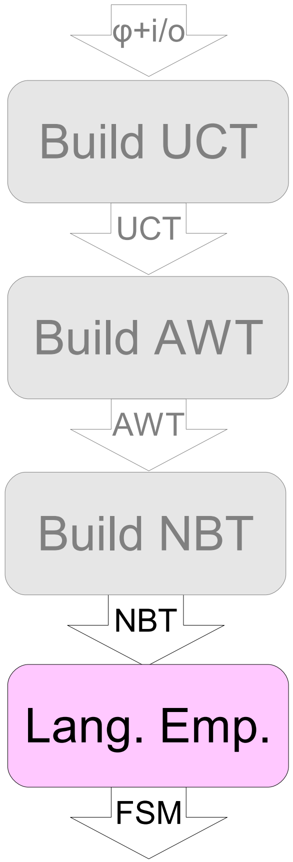
UCT
AWT
NBT
LE

Simplify NBT

- Use simulation relation from AWT
  - cf. Fritz (CIAA'03)
- Idea
  - NBT-state ≈ set of AWT-states
  - Language is conjunct of all sub-language
- Simplifications
  - Two AWT-states A,B in set (NBT-state), B simulates A → B not necessary
  - Two NBT-states S0,S1, if
    - Forall A in S0
    - Exists B in S1 such that A is simulated by B,

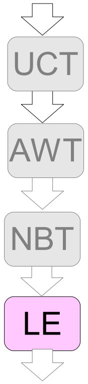
    Then S0 is simulated by S1
  - NBT-states S3, edges to S0 and S1 → construct only edge to S1

Safraless Approach



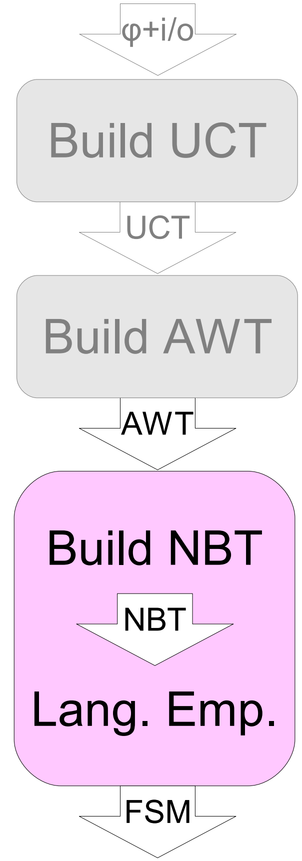
Language Emptiness

- Language Emptiness of tree automata is a game
  - Gurevich, Harrington (STOC'82)
- For NBT - Simple Büchi game
  - System picks: Label and transitions
  - Environment picks: direction
  - Winning condition: visit accepting states infinitely often
- Outcome
  - Game lost ≈ not realizable or
  - Winning strategy ≈ (regular) Σ-labeled D-tree ≈ FSM



Combine MH + Language Emptiness

- Incremental MH
- Idea
  - We just need one witness
  - Expanded NBT in a breadth-first manner
  - Compute language emptiness on expanded part:
    If witness found → Finished
    else proceed with expanding

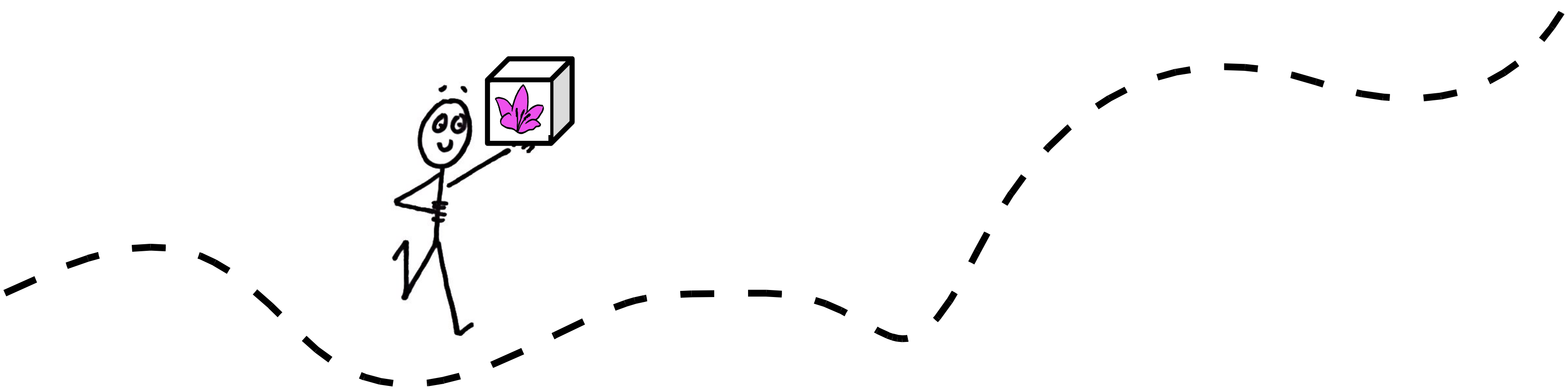


Lily - Linear Logic sYnthesizer

- First tool to offer synthesis for full LTL
- Based on Fabio Somenzi's Wring
- Implements all mentioned optimizations
- http://www.ist.tugraz.at/staff/jobstmann/lily/

Conclusion

- Still a long way
- A nice toy to play with
- Example are small but useful for property debugging (or learning LTL)
- Optimizations are enabling factor
- Future?



Thank you for your attention!