Tiling and Walking
automata on trees and grids

Hendrik Jan Hoogeboom
Leiden

Workshop on
Non-Classical Models of Automata and Applications
August 31 - September 1, 2009, Wroclaw, Poland

themes

acceptance

- tiling $\mapsto$ global labelling
- walking $\mapsto$ local control

structures

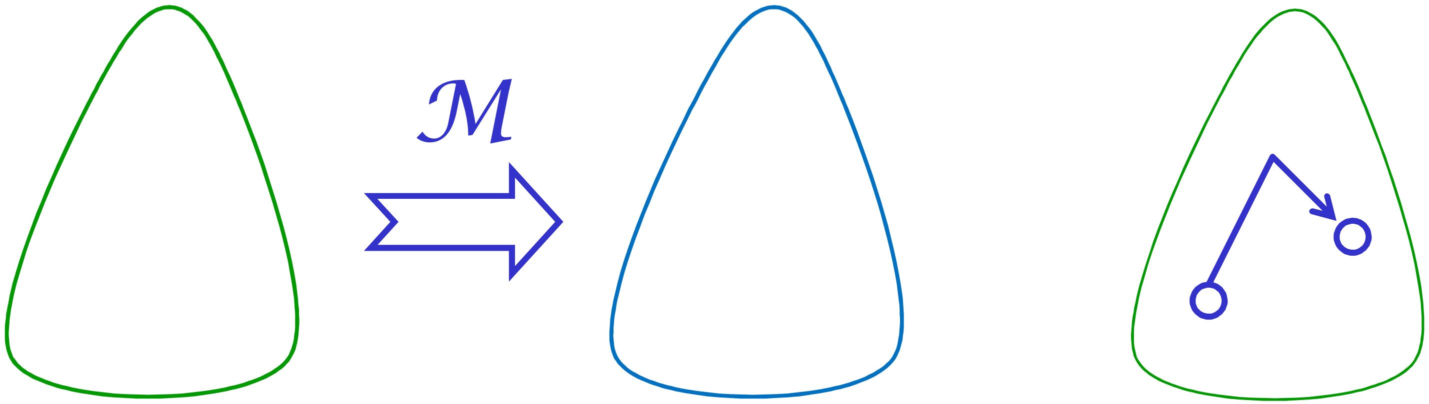
- strings
- trees
- pictures
- …

pebbles

- *non-classical*

background

- XML document transformation

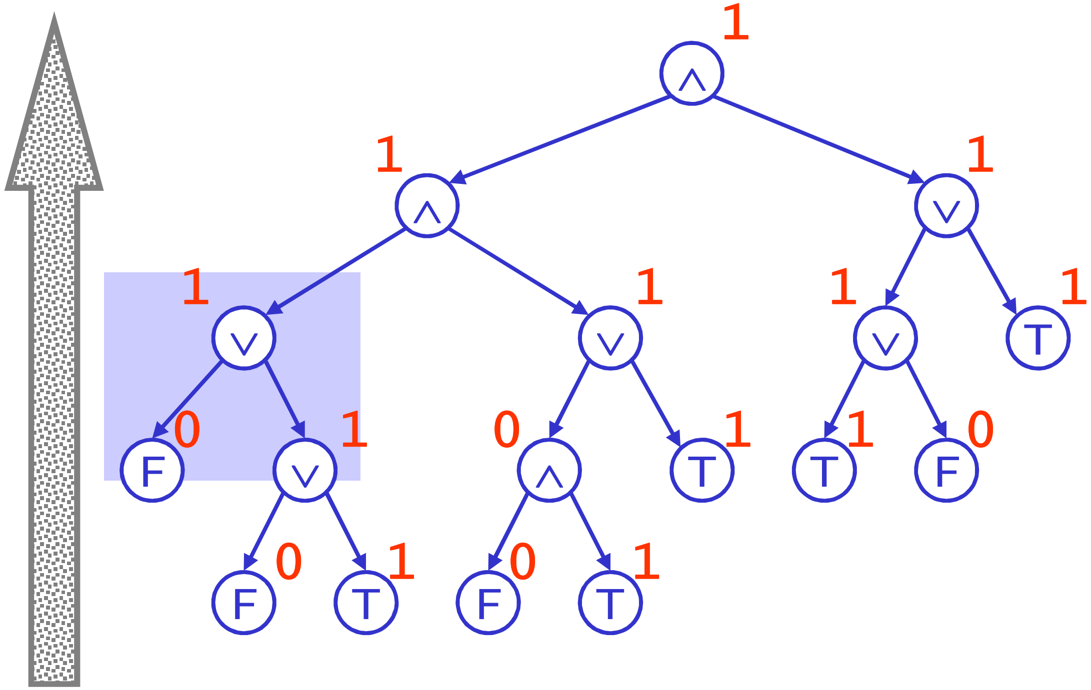


- graph exploration

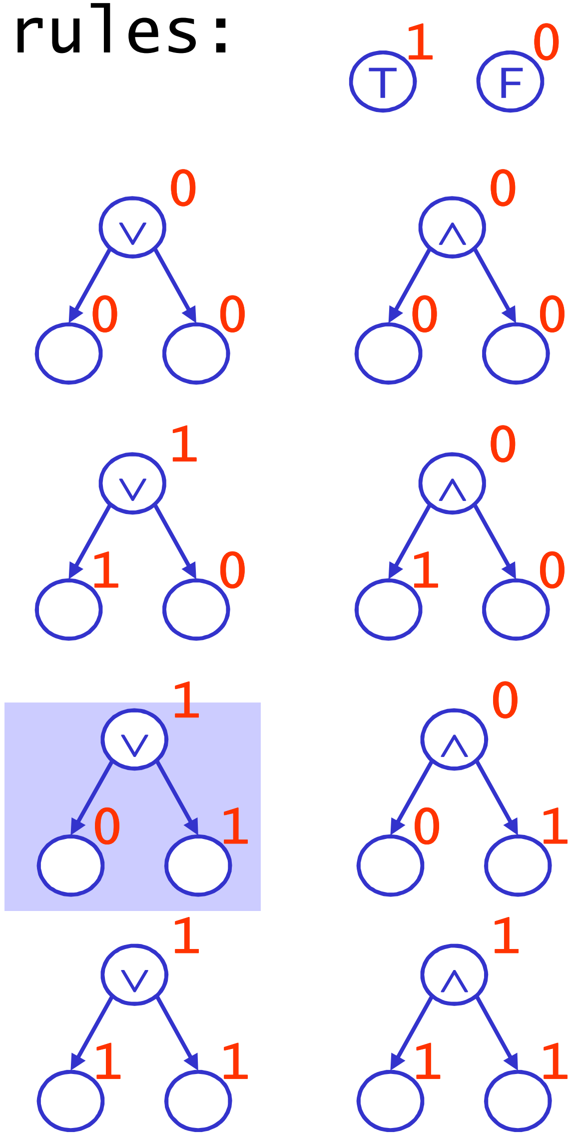
  many heads on graphs 'robots'

  grids, toruses, mazes, …

bottom-up tree automaton

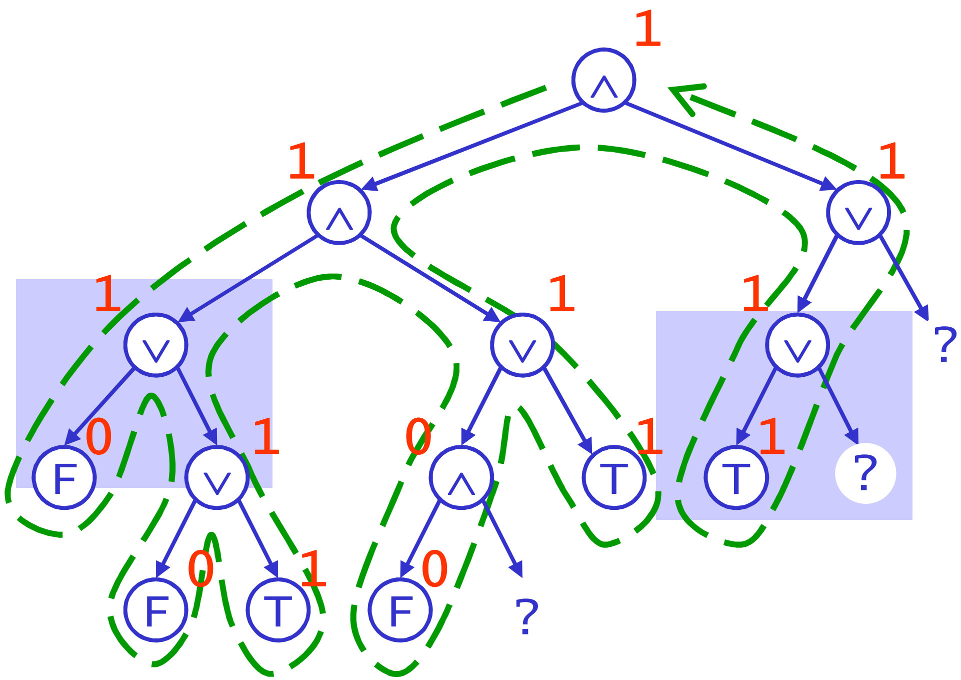


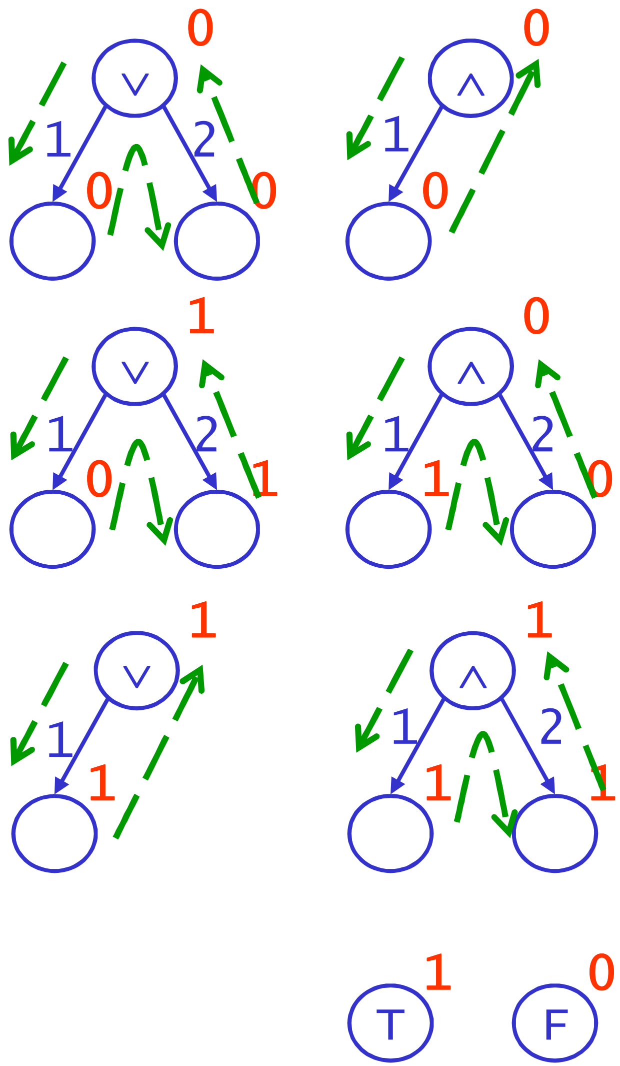
rules:



bottom-up evaluation

walking along the tree





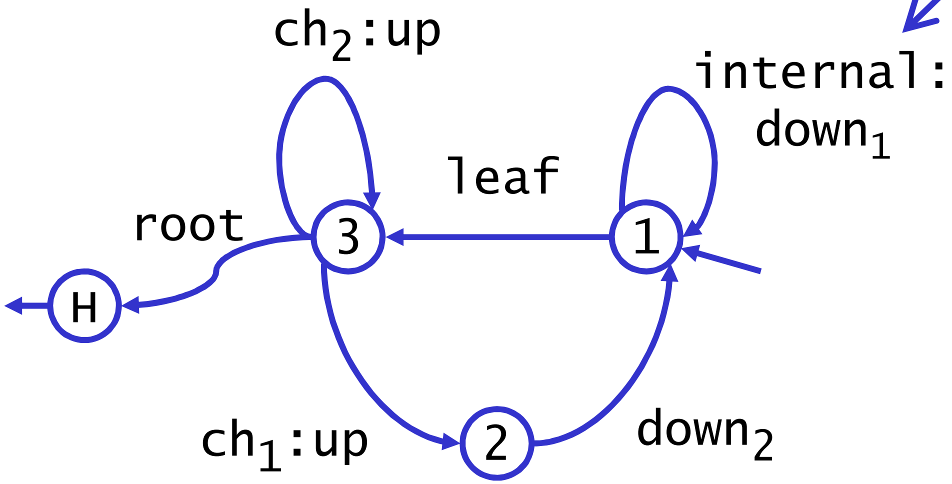
evaluates and/or trees !

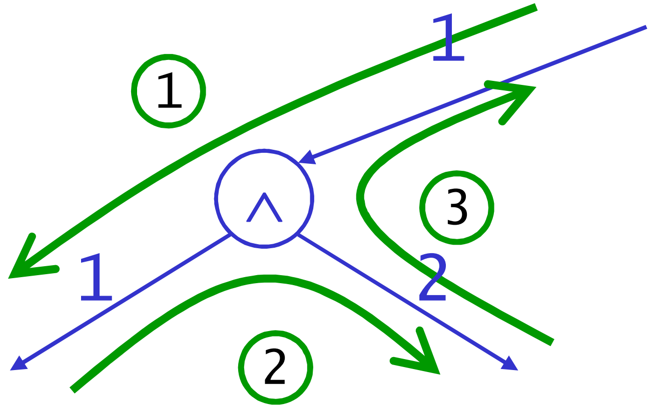
cf. two-way finite state automaton

tree walking automaton

example: complete pre order tree traversal

at internal node move to first child

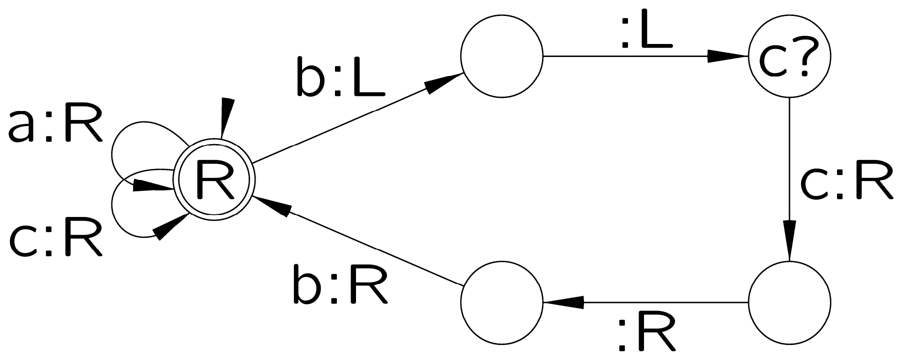




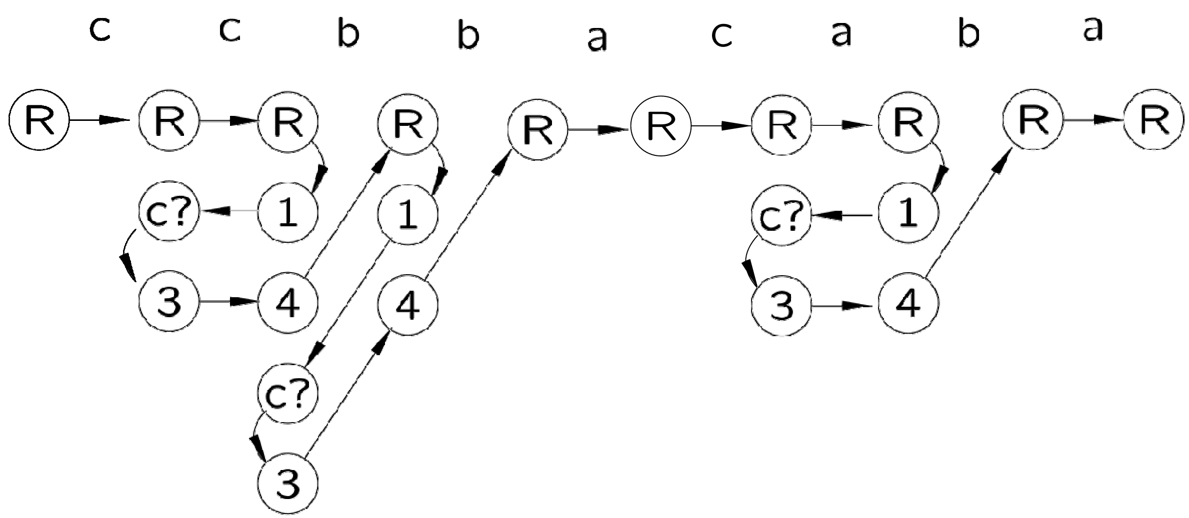
walk along edges, moves based on

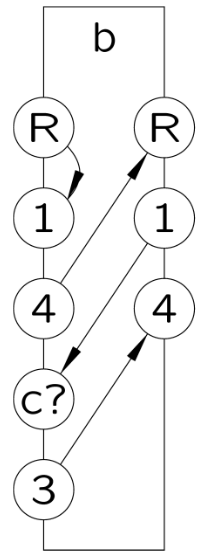
- state (2)
- node label lab
- child number ch
  (= incoming edge)

strings: 2way automata



‘visit sequences’

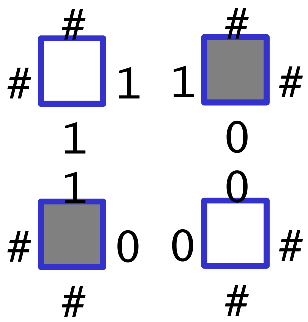




Rabin&Scott, Shepherdson (1959)

pictures (arrays, lattices)

$(\Sigma, \Gamma, T, \#, \varphi)$	tiling system
Σ, Γ	tile, edge colours
T	tiles with 4-sided markings
$\# \in \Gamma$	border colour
$\varphi : T \to \Sigma$	tile $\mapsto$ colour



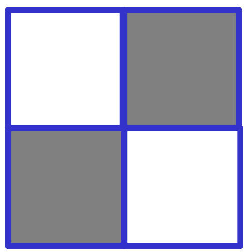
tiling: Γ-markings match

language: Σ-labelled rectangles

TILE

recognizable REC

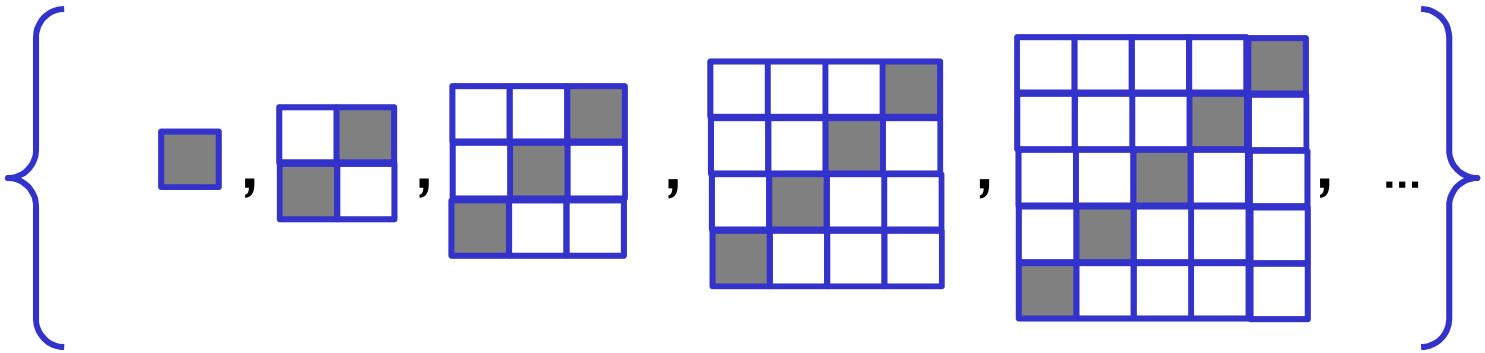
(Giammarresi & Restivo, Handbook FLT 1997)



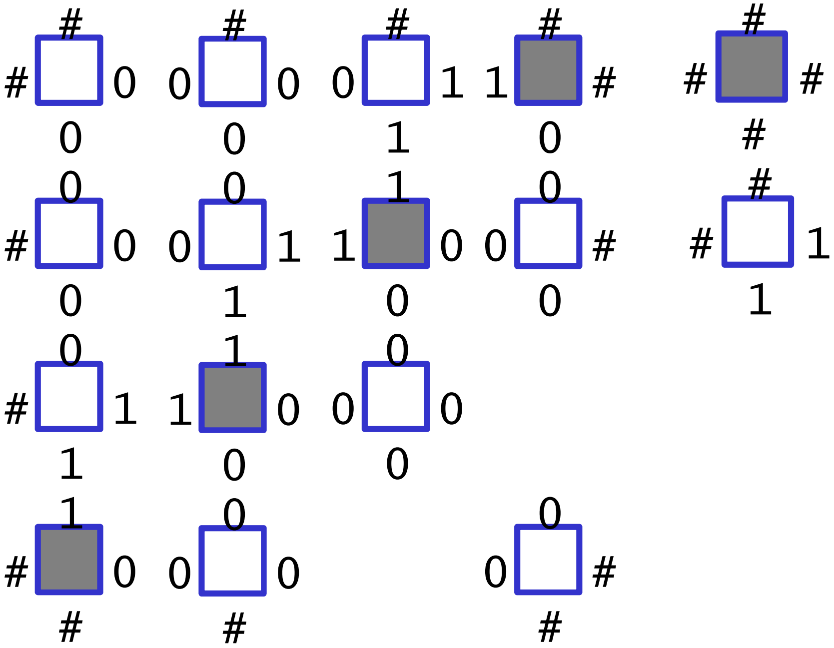
Wang tiles

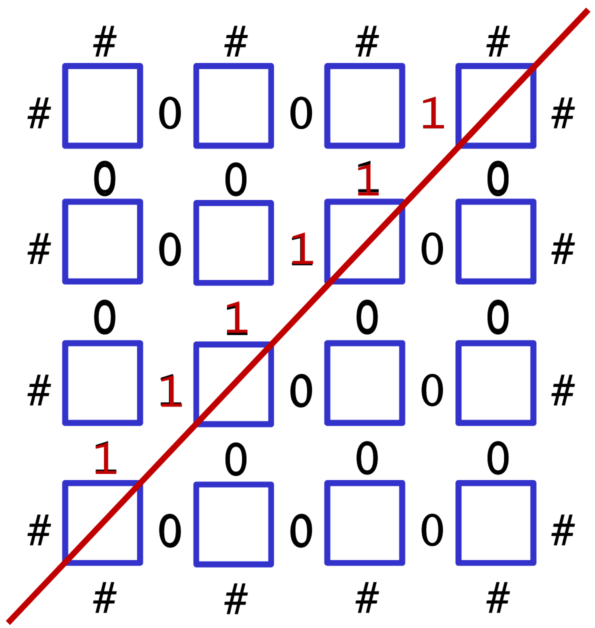
local lattice languages h(LLL)

squares ∈ TILE



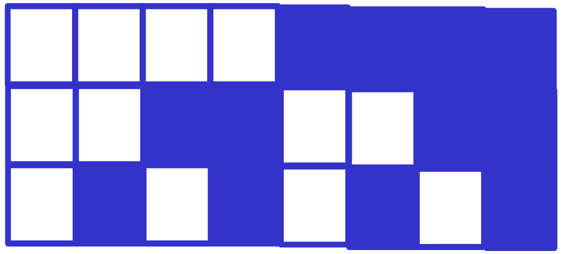
may add more colours …



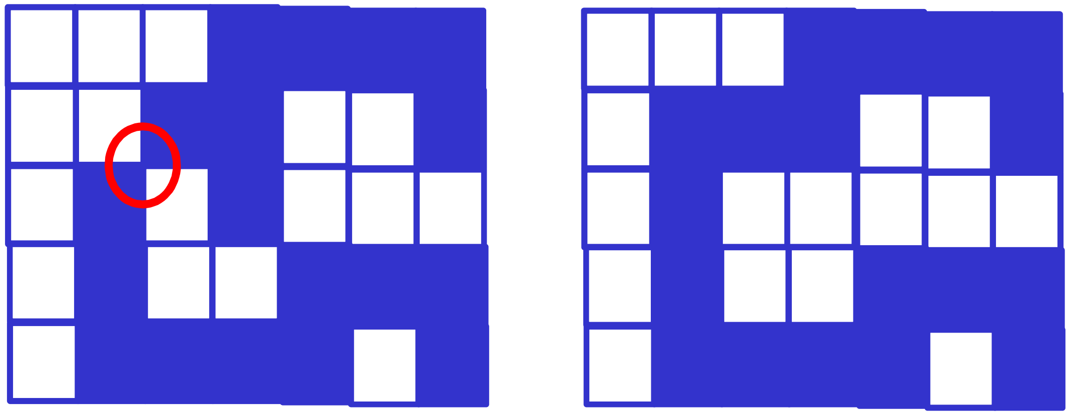


other languages in TILE over pictures

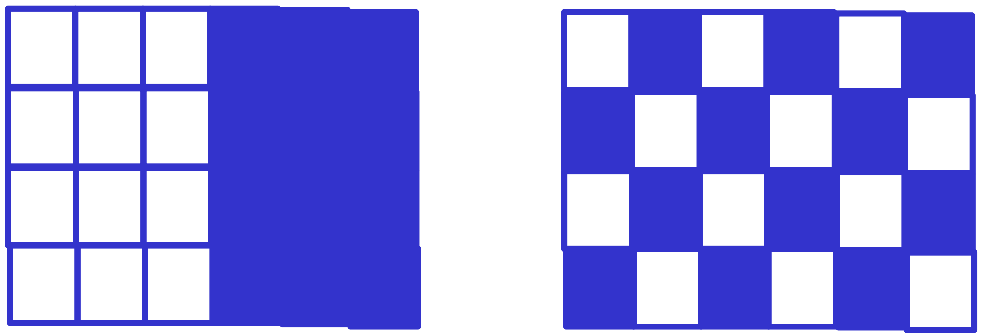
- binary counting



- connected area (*)



- equal numbers (**)

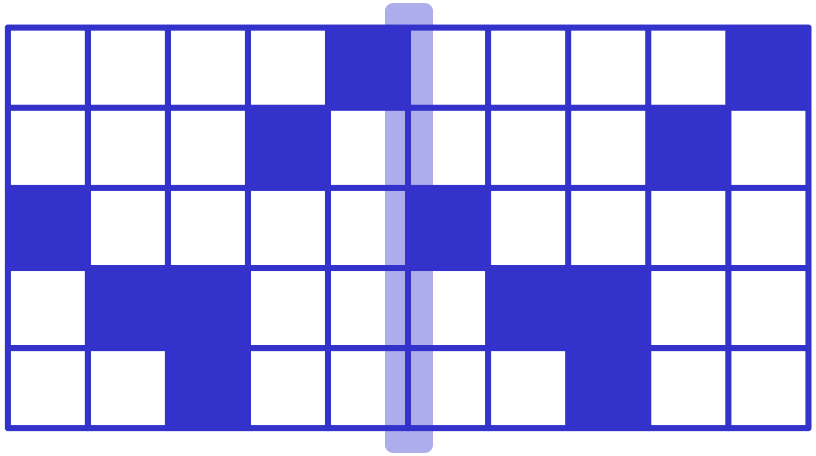


(Reinhardt)

not closed complement

$$\{ww \mid w \in \{a,b\}^*\}$$

equal-□□ $\notin$ TILE

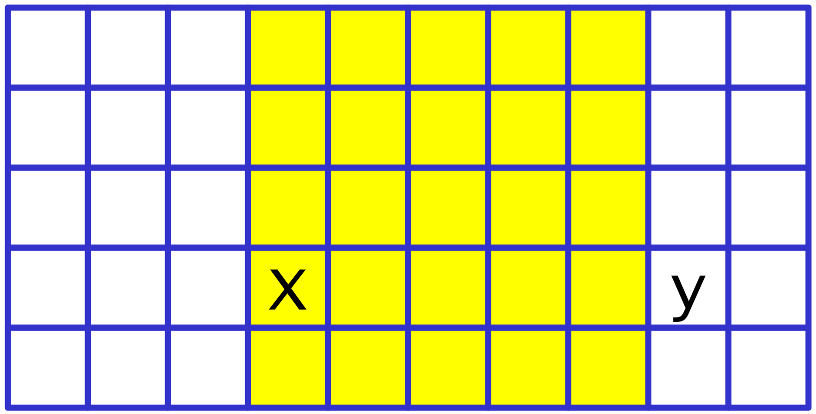


c tile colours

e edge colours

$c^{n^2} > e^n$

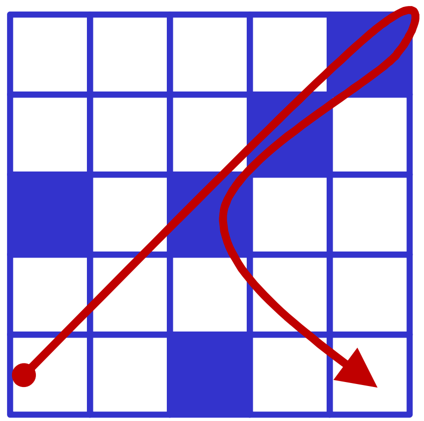
unequal-□□ $\in$ TILE



(Kari & Moore, STACS 2001)

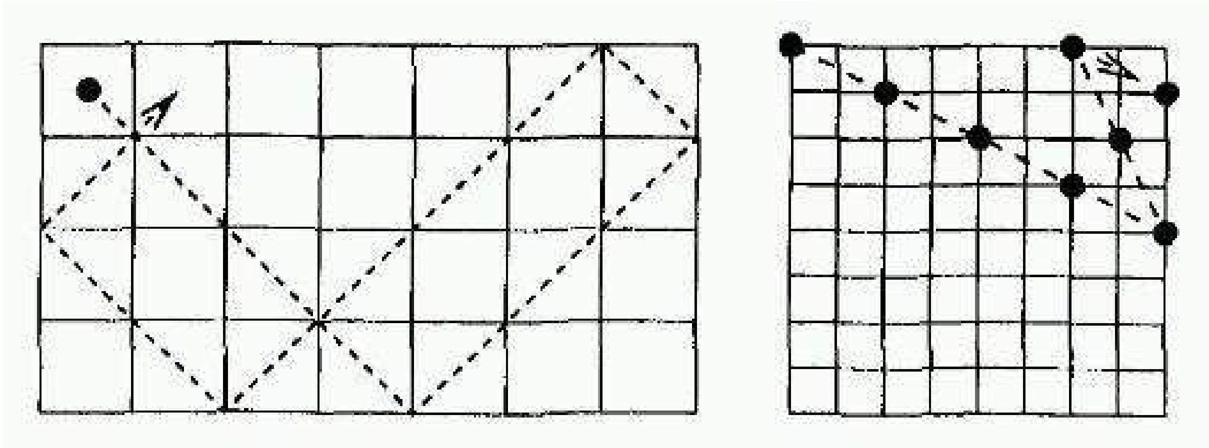
‘four-way automaton’

odd squares:
middle cell blue



nondeterministic

examples

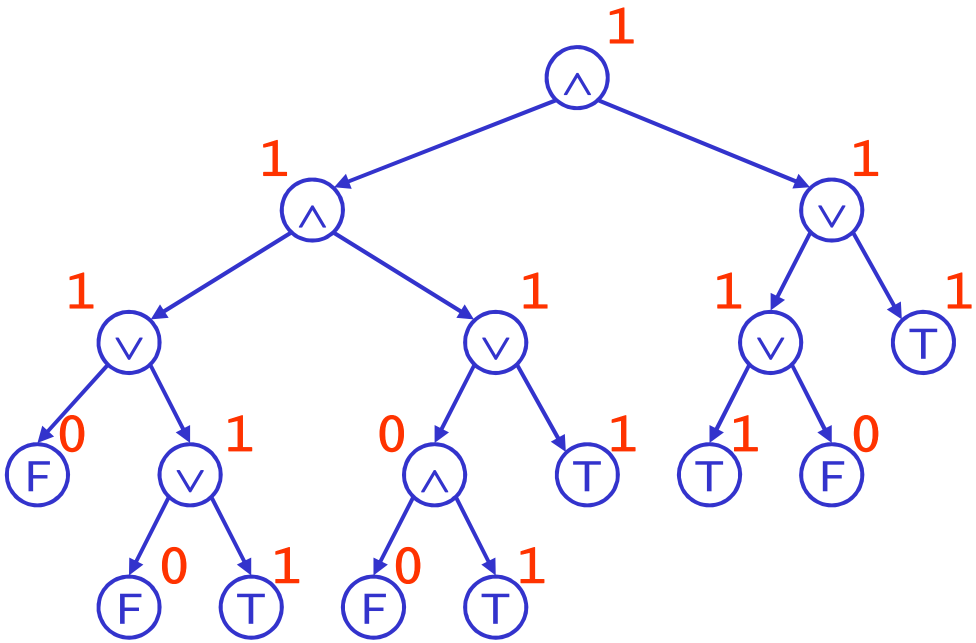


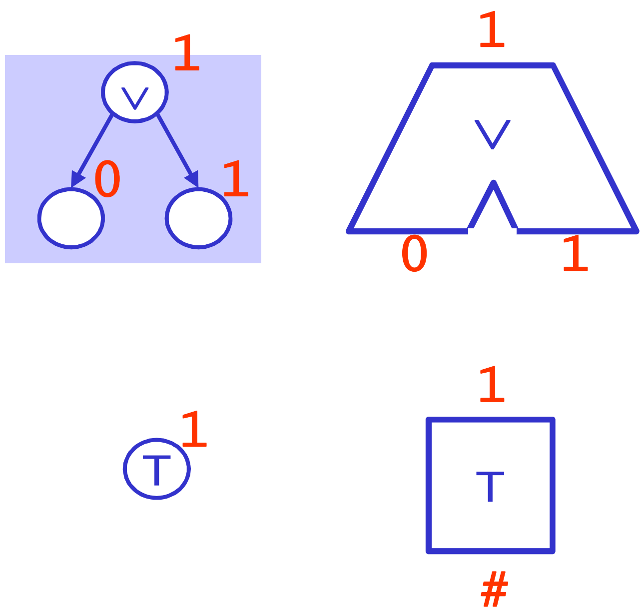
"By bouncing like a billiard ball or making knights' moves, and ending one cell from the corner, a DFA can check that the two sides of a rectangle are mutually prime, or that the side of a square is a power of 2"

avoid infinite loops!

(Lindgren, Moore & Nordahl, J. Stat. Phys. 1998)

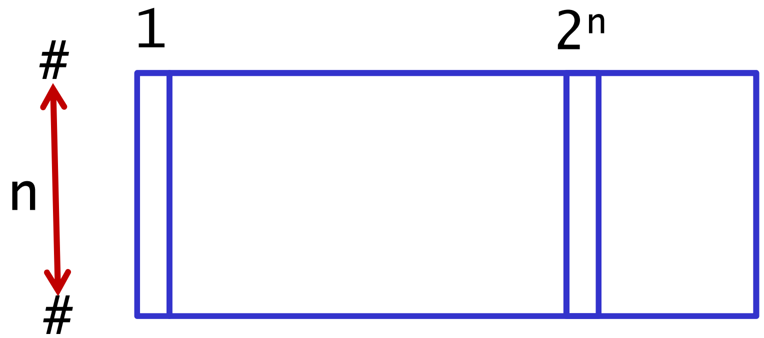
tree automata as tiling system





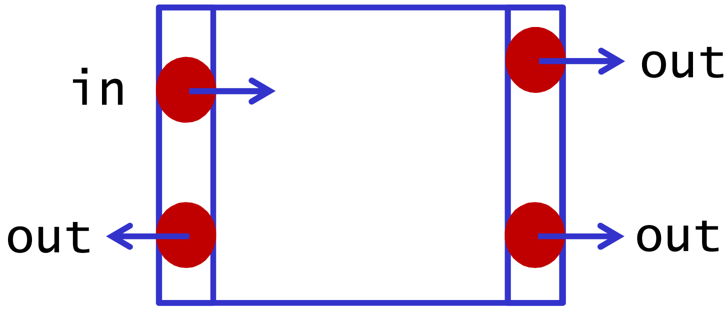
not accepted by walking

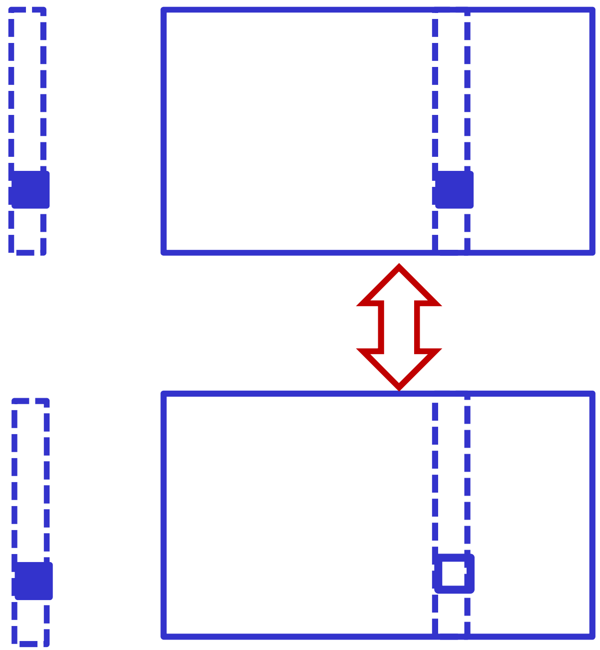
(Inoue & Nakamura, 1977)



equivalent $n \times 2^n$ ‘chunks’ (k states)

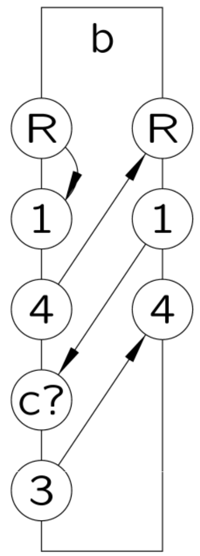
$$(2^{2(n+2)k+1})^{2(n+2)k} < 2^{(n2^n)}$$





TILE vs. WALK

WALK ⊆ TILE



visit sequences

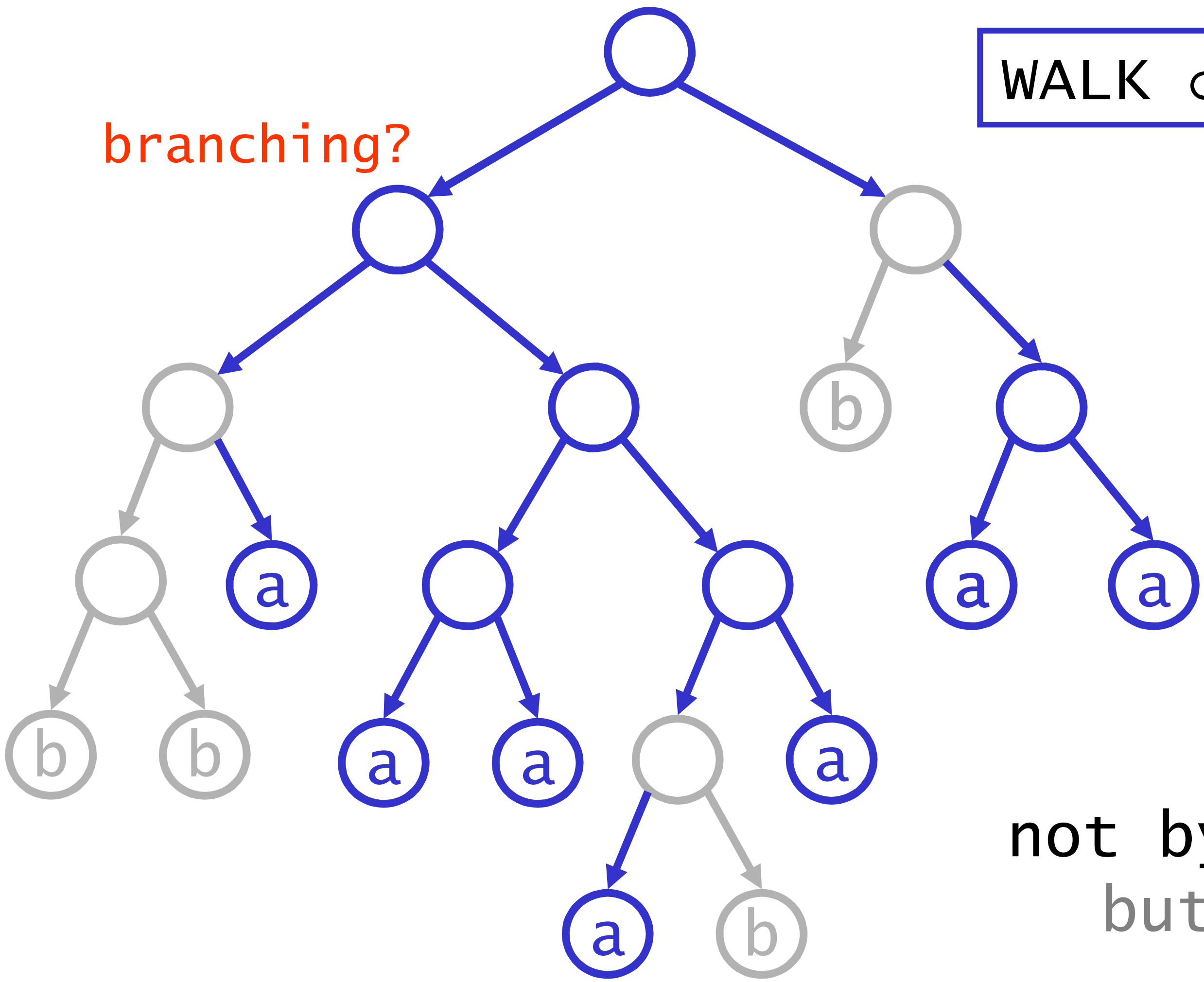
WALK = TILE (REG) strings

WALK ⊂ TILE (REG) trees (open untill 2005!)

WALK ⊂ TILE (REC) pictures ‘chunks’

‘branching structure’ of even length

Bojańczyk & Colcombet STOC, 2005



WALK ⊂ TILE

for trees

not by WALK

but FO

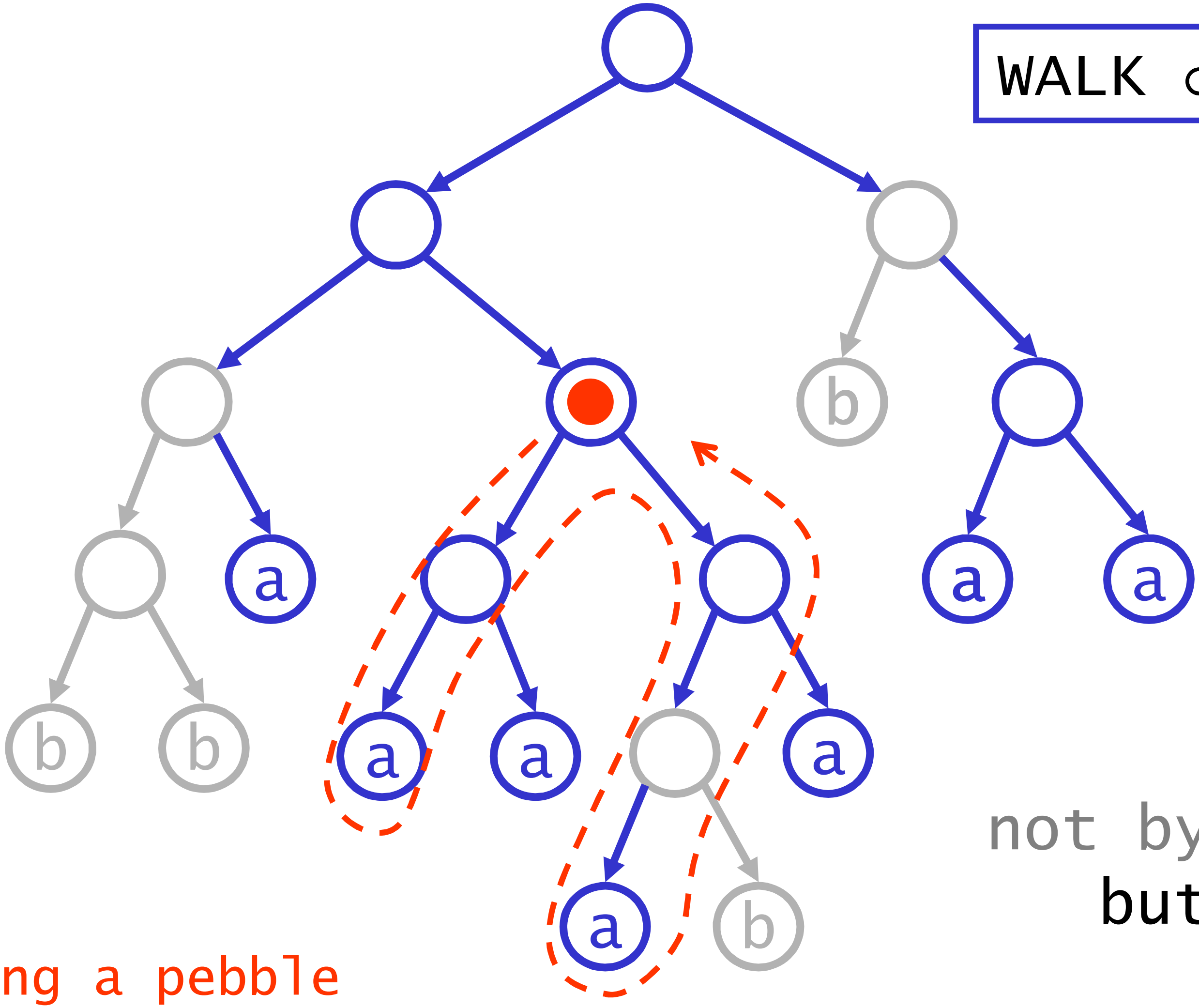
ADDING PEBBLES

pebbles to mark nodes

Bojańczyk & Colcombet

WALK ⊂ TILE

for trees



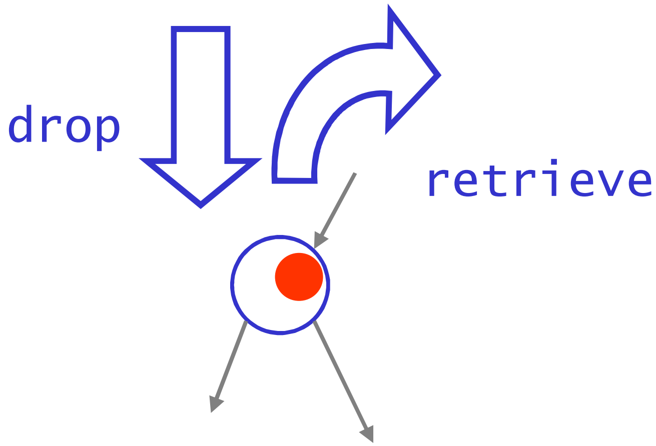
not by WALK

but PBBL

using a pebble
to determine branching

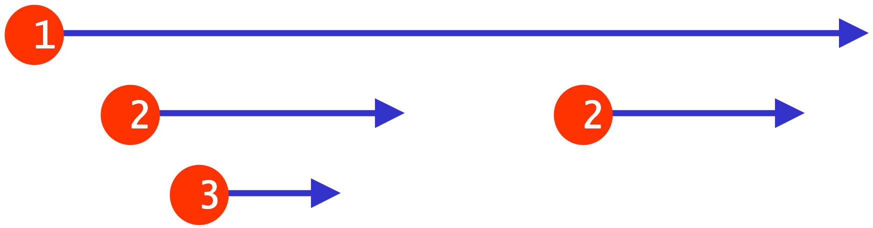
adding nested pebbles to the TWA

pebble: mark a node



- fixed number for automaton
- can be distinguished & reused
- used to determine where to go

- *nested lifetimes* ‘stack discipline’

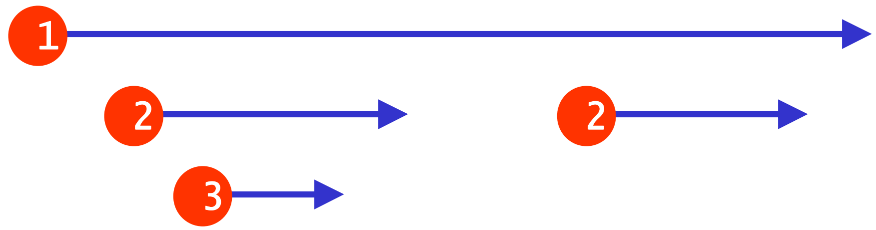


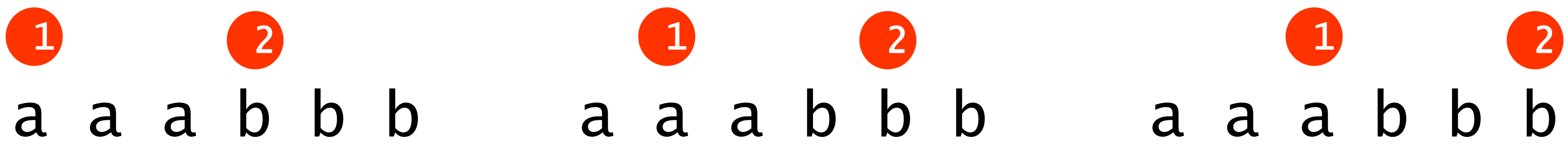
‘regular’ extension

for trees

beware of the pebble

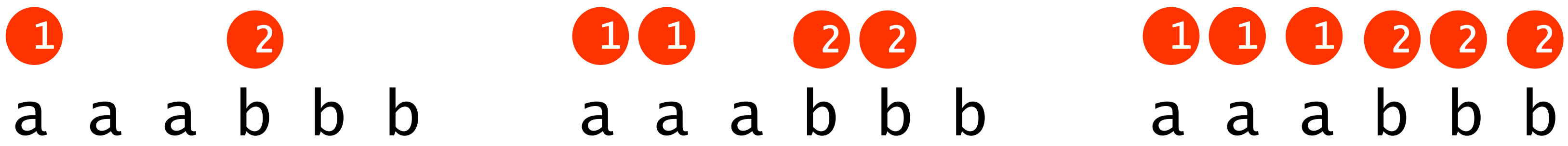
avoid counting





general: LOGSPACE

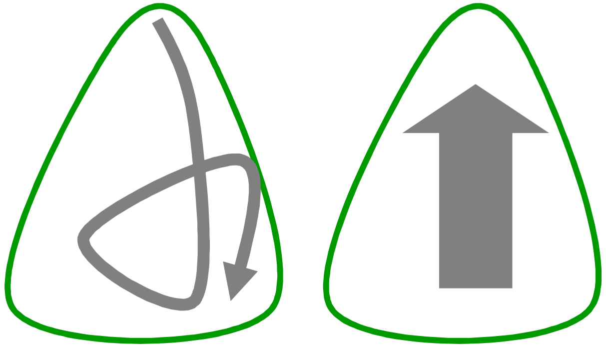
▶ nest!



▶ bounded number!

power of tree walking automata

pebble



WALK $\subseteq$ REG (≡TILE)

WALK $\subset$ REG

Bojańczyk & Colcombet STOC'05

PBBL $\subseteq$ REG

Engelfriet & H '99

PBBL $\subset$ REG

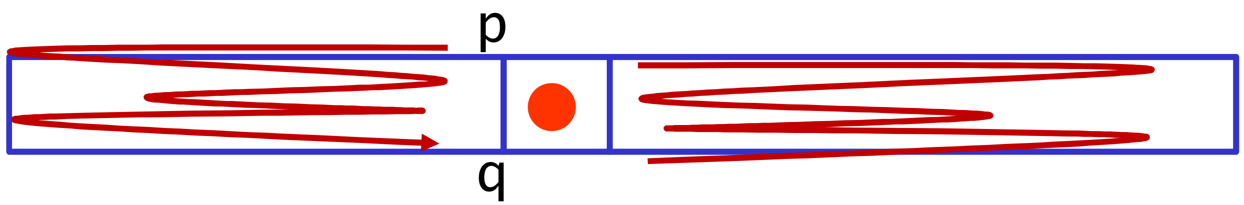
Bojańczyk, Samuelides,
Schwentick & Segoufin ICALP'06

"tree walking automata easily loose their way"
(even with the help of pebbles)

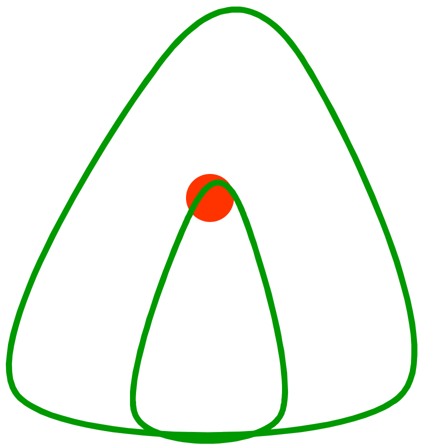
nested pebbles on strings

single pebble

keep track of excursions



similar techniques in trees

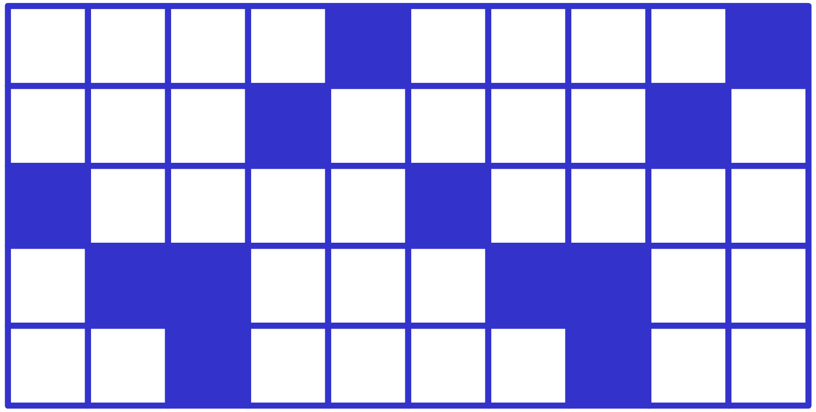


PBBL = WALK = TILE
for strings

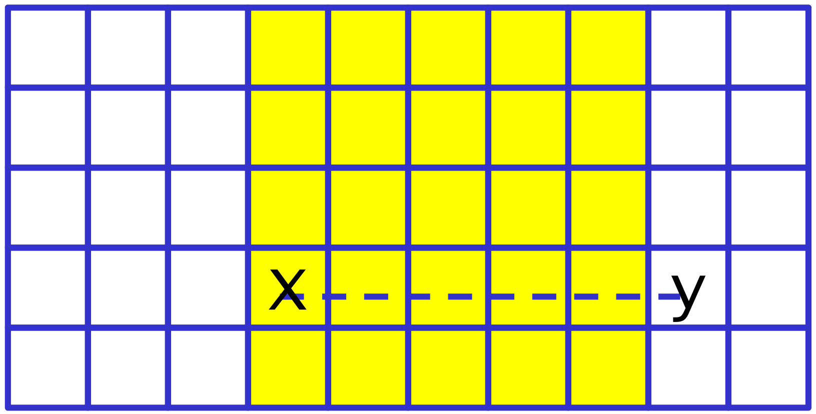
single pebble on pictures

equal-□□ $\notin$ TILE

unequal-□□ $\in$ TILE



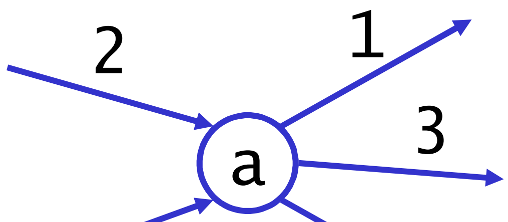
(un)equal-□□ $\in$ 1PBBL



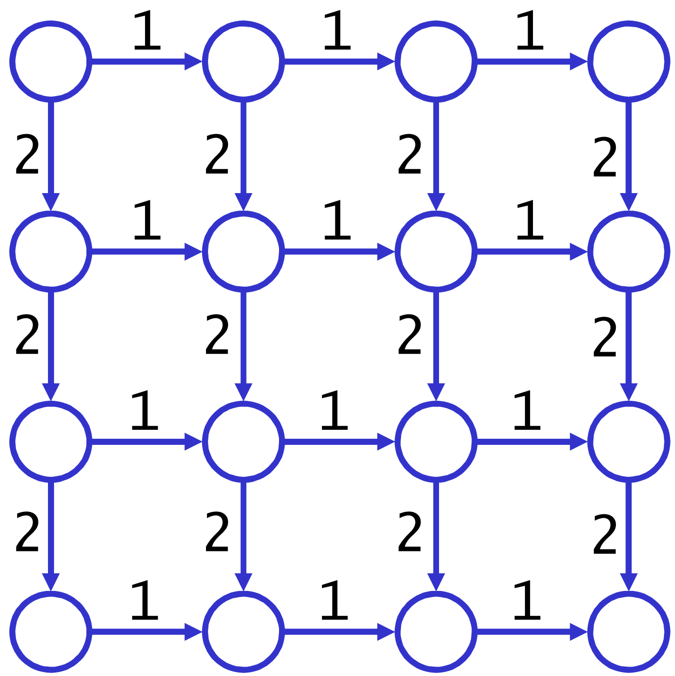
(cf. Nakamura, ‘one-pebble rectangular array acceptors’, IPL, 1981)

CONNECTIONS TO LOGIC

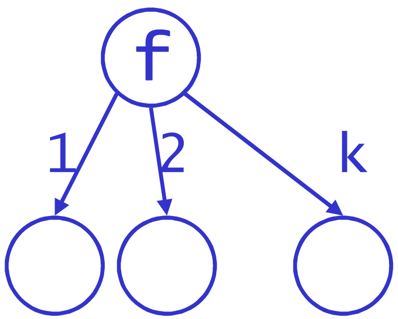
labelled graphs



edges:
locally injective



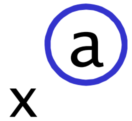
picture=grid, torus



trees

mso logic

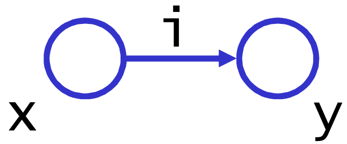
x, y	node variables
X, Y	node sets



$lab_a(x)$

$edg_i(x,y)$

$x \le y$ (partial order)



$x = y$

$x \in X$

$\neg \ \wedge \ \vee$	connectives
$\forall x \ \exists x$	quantification (nodes)
$\forall X \ \exists X$	(sets)

FO	first order
EMSO	existential mso

example (strings)

next a

no position in between has a

number of a’s is even

there exists a set X that contains

every other position with a

b a a b a b b a a b b a

closed formula $\mapsto$ language

for strings, trees

MSO = REG (aka TILE)

for pictures

EMSO = REC (aka TILE)

graph acceptors

(Thomas: Logics, tilings and automata, ICALP 1991)

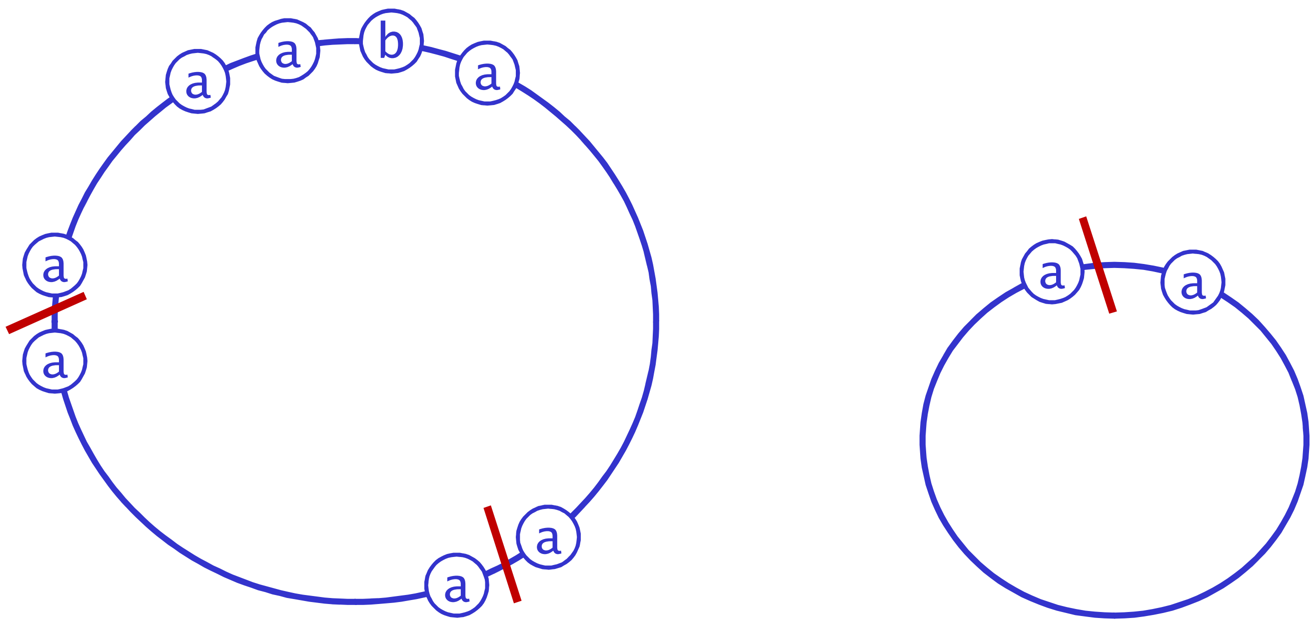
- labelling of nodes with states
- look at subgraphs of certain diameter
- distinguish types
- acceptance: boolean combination of
  ‘type α occurs $\leq$ k times’ FO

► generalization of TILE
corresponds to EMSO

diameter and counting can be avoided in special cases

tiling necklaces

‘has label b’ $\notin$ TILE



graph acceptors can force ‘b’-tile

FO+dTC ⊆ dPBBL

(1) logic to nested pebbles

$\varphi \rightarrow \mathcal{A}$

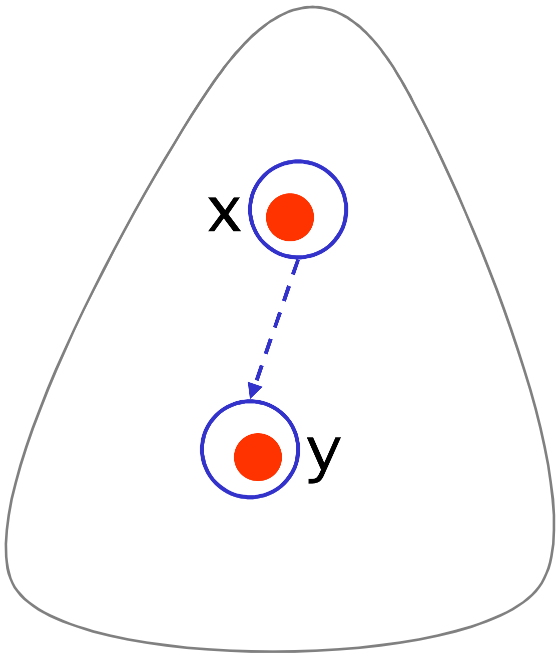
$lab_a(x)$
$edg_i(x,y)$

free variables ~ fixed pebbles

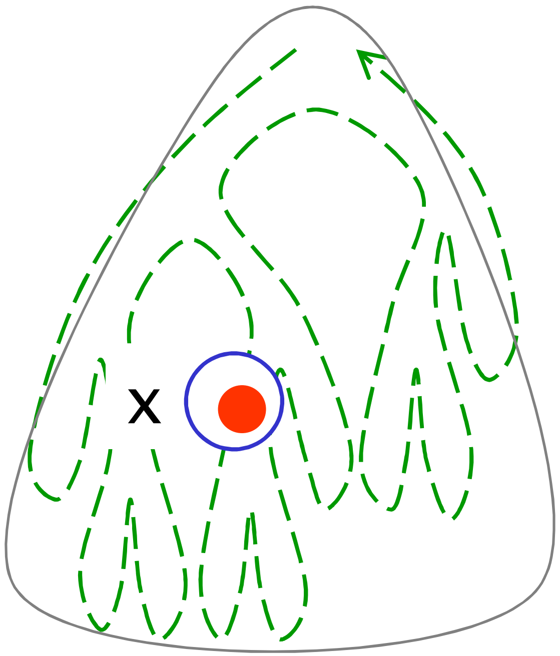
$x \leq y$
$x = y$

¬ ∧ ∨
∀x ∃x

$\varphi^*(x,y)$



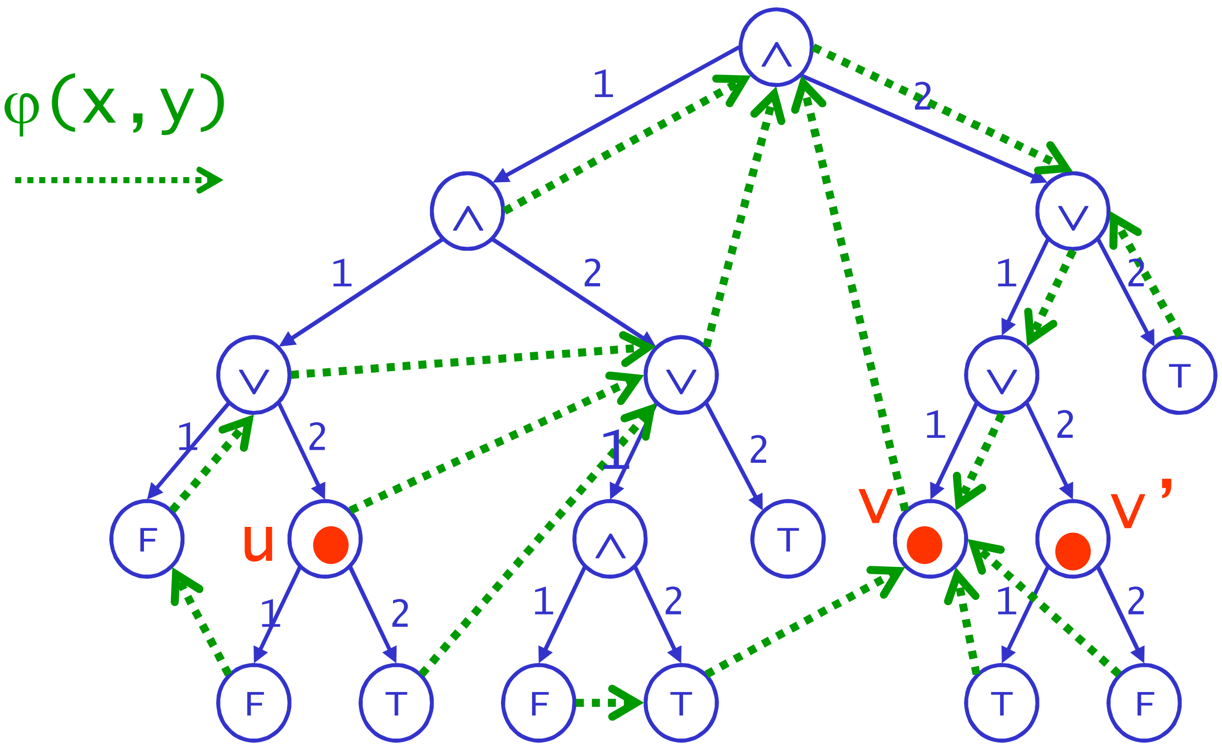
$x \leq y$



$\forall x \ \varphi(x)$ $\mathcal{A}_\varphi$

transitive closure

$\varphi(x,y)$



$\varphi^*(u,v)$ tc

deterministic tc: φ functional

$\varphi(u,v,z)$

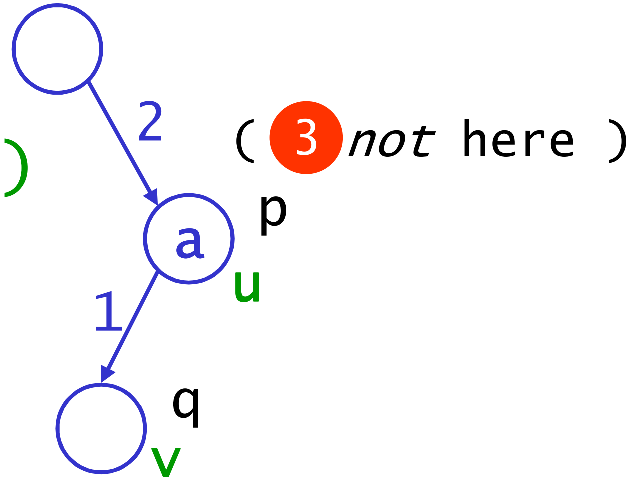
dPBBL $\subseteq$ FO+dTC

(2) nested pebbles to logic

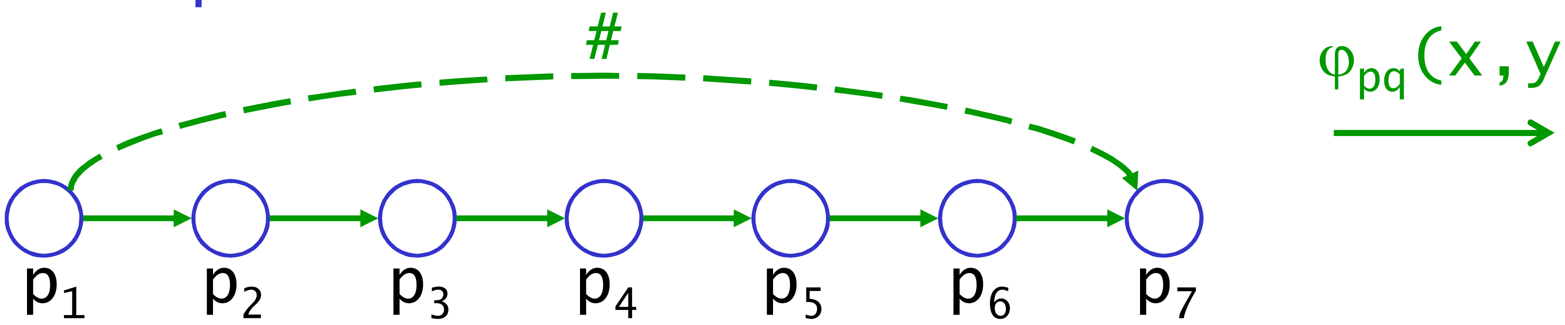
i single move $\varphi_{pq}(u,v)$

$$lab_a(u) \wedge (\exists u')edg_2(u',u)$$
$$\wedge\ u \neq x_3 \quad \wedge \quad edg_1(u,v)$$

free variables for pebbles



ii computation ~ tc with states



Kleene: removing states finite aut to reg expr

iii dropping pebbles ~ tc (inductively)

for general graph-like structures

TILE = EMSO

PBBL = FO+TC

(almost)

use a guide

FO+dTC = dPBBL
FO+posTC = PBBL

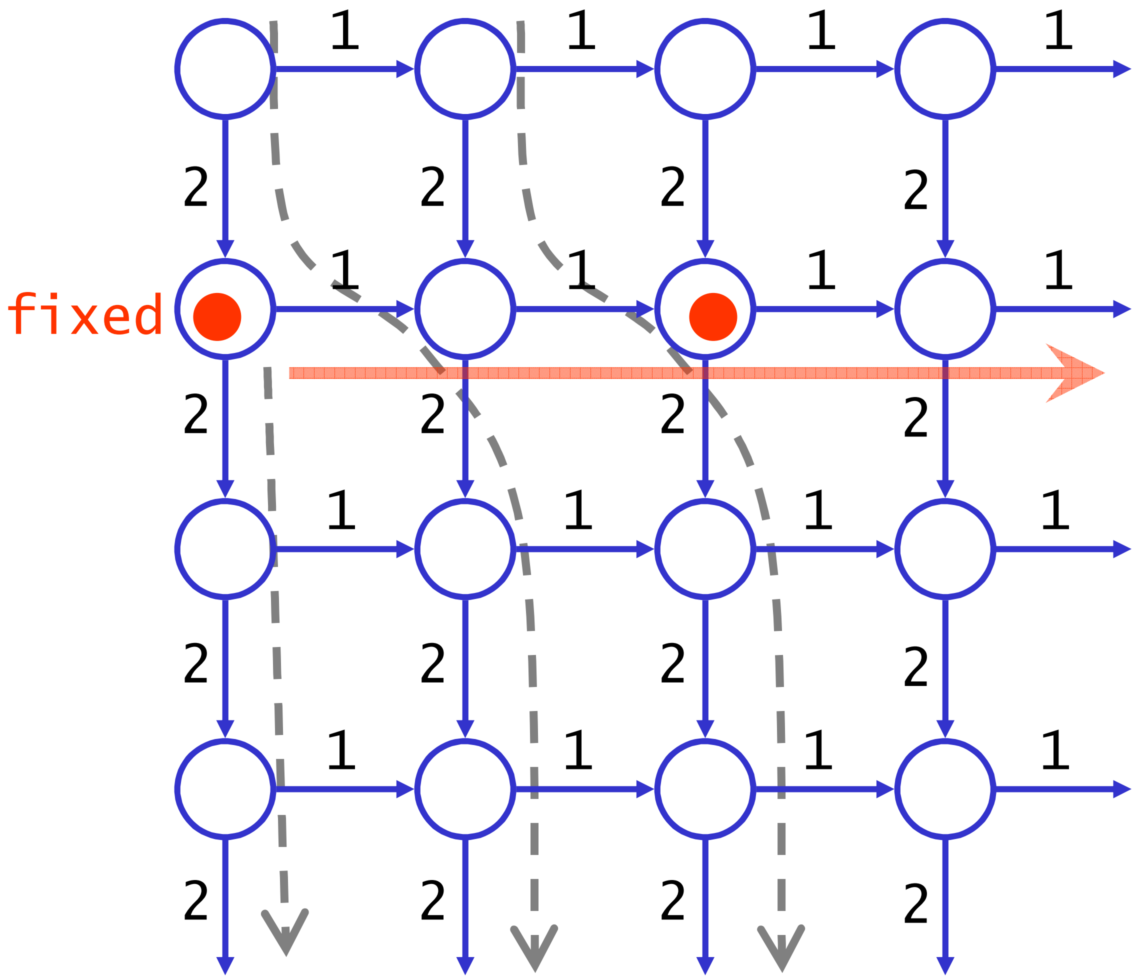
for families of *searchable* graphs
with a ‘guide’

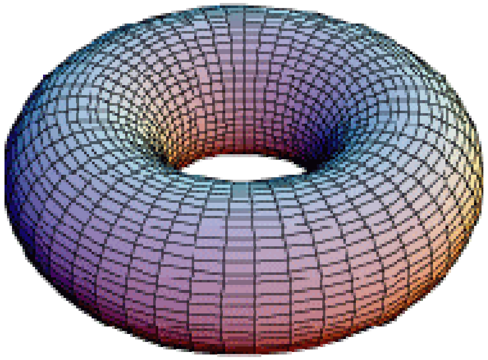
guide: deterministic
with pebbles
visits each node (at least) once
& halts

$(\forall x)\ lab_0(x)$

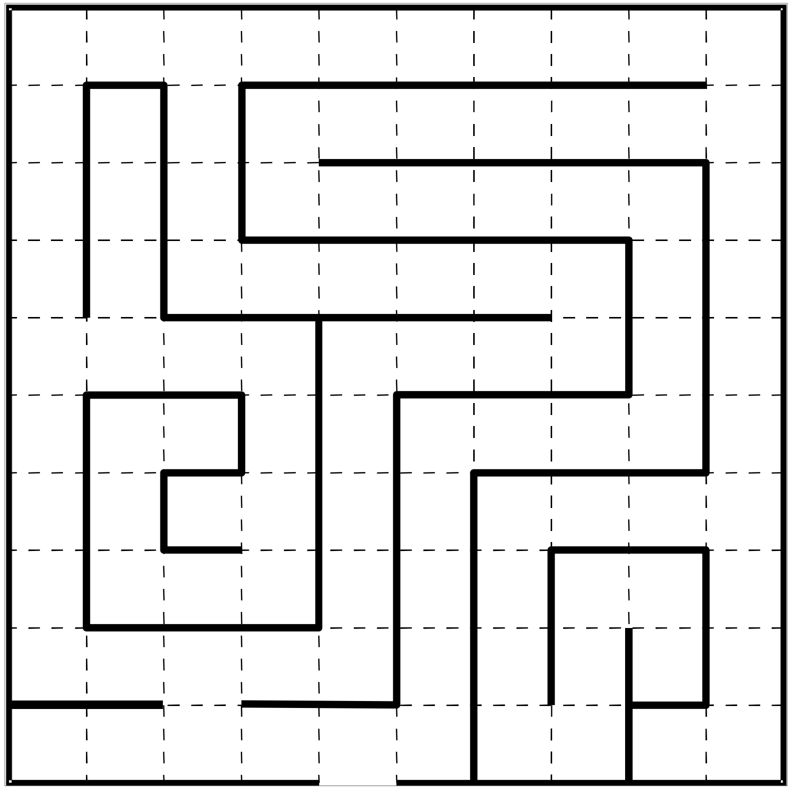
unranked trees, grids, toruses, …
2 pebbles

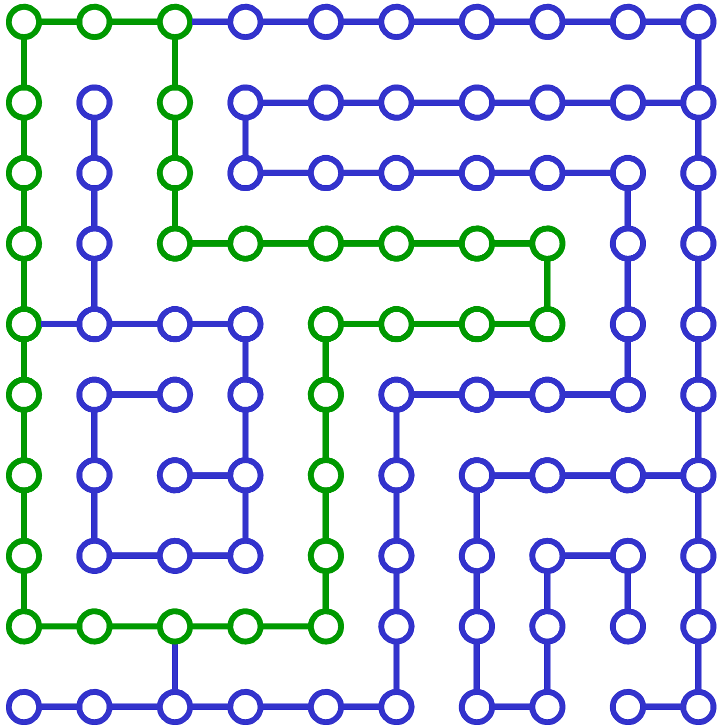
walking the torus



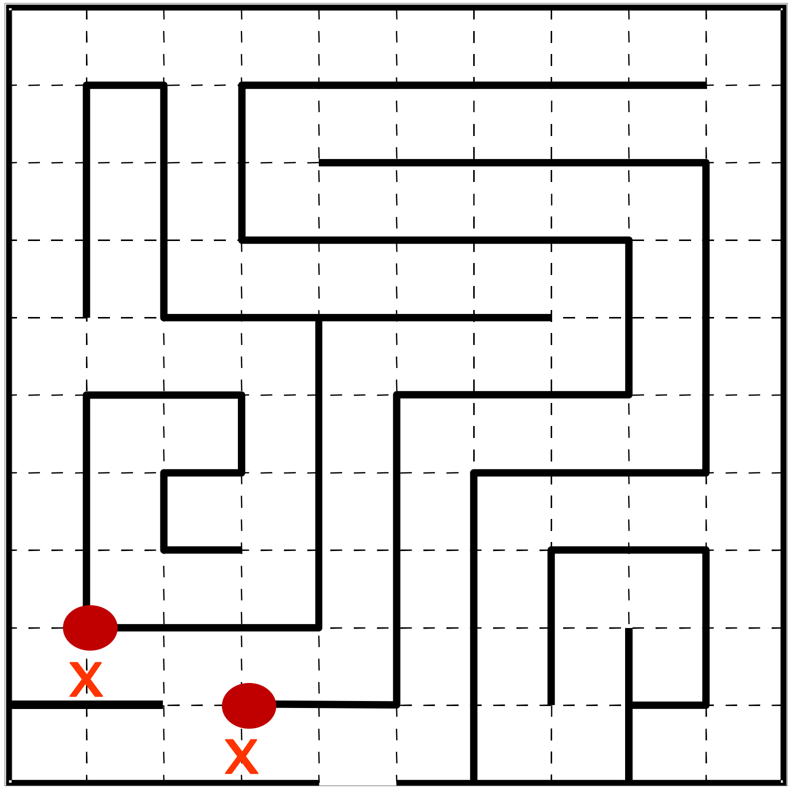


two pebbles
(nested)





mazes



(Blum & Kozen 'power of the compass')

counter, or
two heads, or
two pebbles
(*not* nested)

single pebble not ok

(Budach)

‘invisible’ pebbles obtain *all* REG trees
(distributed stack)

nested pebbles and
… pictures 1PBBL $\not\subseteq$ TILE
… mazes
… traces / dependence graphs
… texts (two orders on string)

strong / weak pebbles


‘tossing Pebbles’
Joost Engelfriet
Hendrik Jan Hoogeboom
Leiden NL
thank you …