Hierarchical Partitioning (Really) Helps

—

From Flat Structures to Hierarchies

Rajeev Alur

University of Pennsylvania

alur@cis.upenn.edu

M. Oliver Möller

BRICS* **Århus**

omoeller@brics.dk

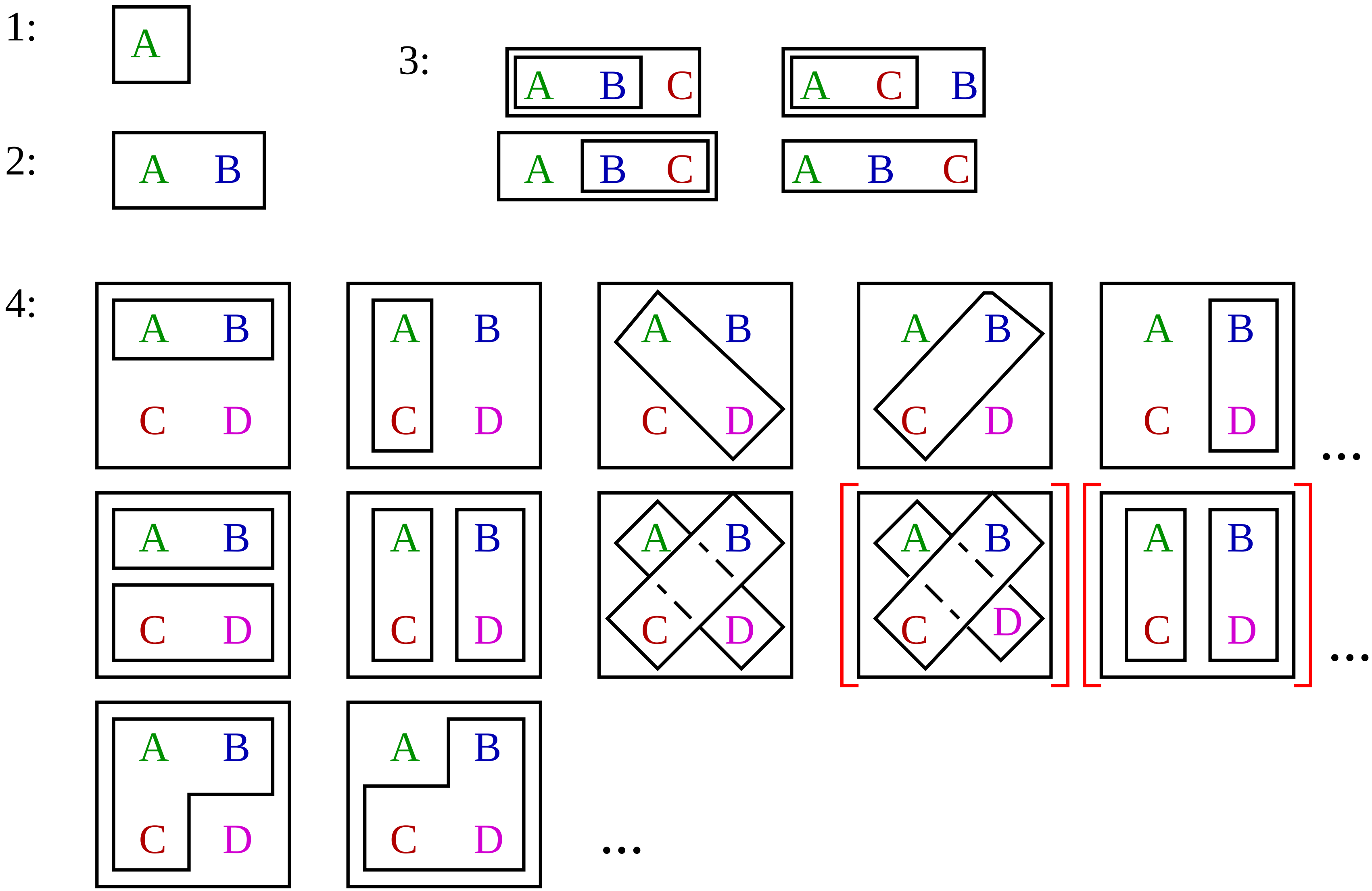
*Basic Research in Computer Science

Order of Multiplication

$$42^{17} \quad \times \quad \sqrt{2} \quad \times \quad 1/\sqrt{3} \quad \times \quad \sqrt{0.0003/2}$$

- Is there a *optimal* way to compute the product?
- If yes, can we *find* it?
- If yes, how *difficult* might that be...?

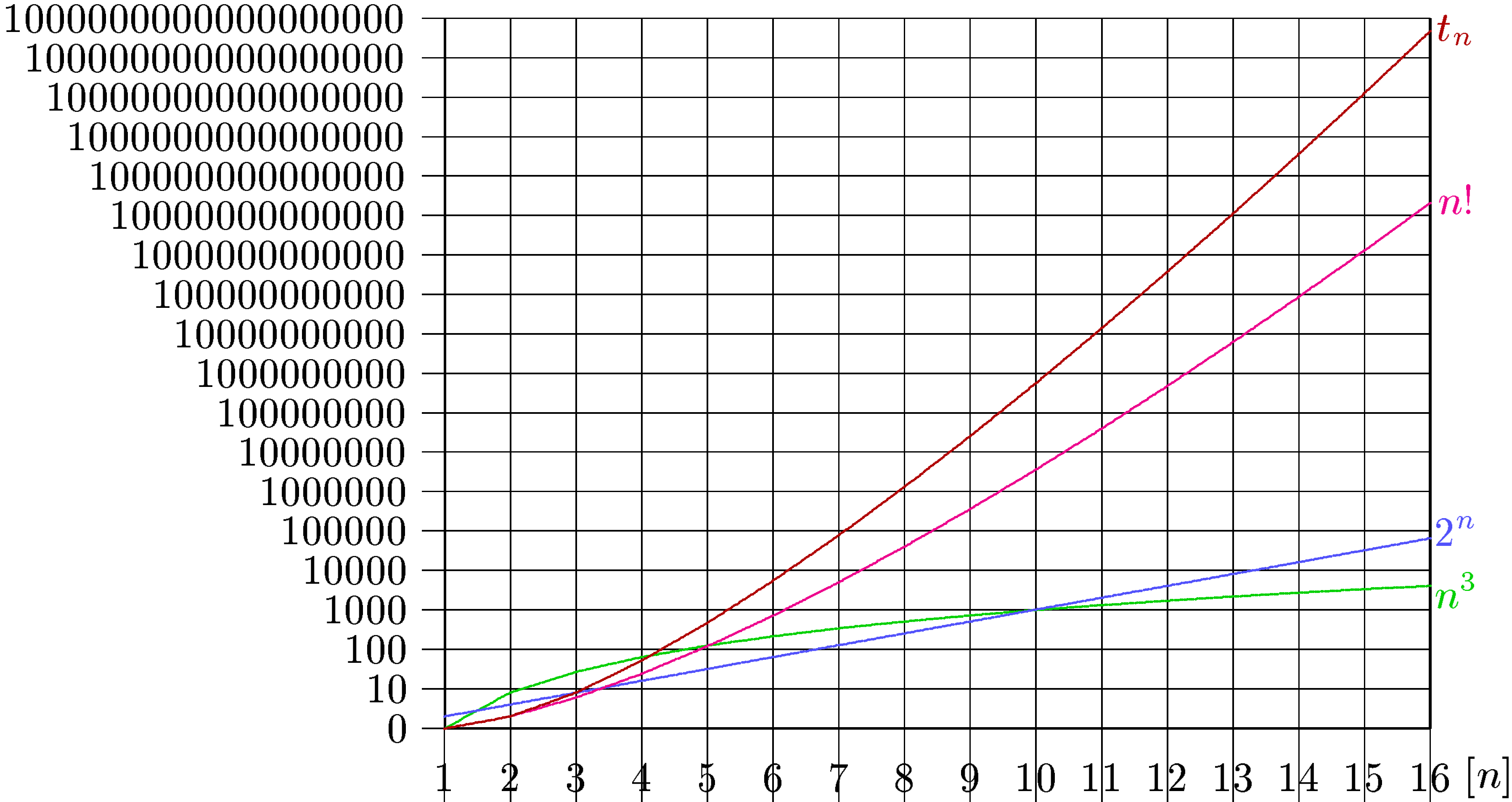
How Many Ways to Group Together?



Four Elements: Already 26 Structures

- the problem was first posed by Ernst Schröder 1870
  [Zentralblatt für Math. Physik, „Vier combinatorische Probleme“]
- we *still* do not know a closed formula
- *but* can compute the number of hierarchical partitionings for fixed n efficiently

Combinatorial Explosion



n : number of components

t_n : number of different hierarchical partitionings

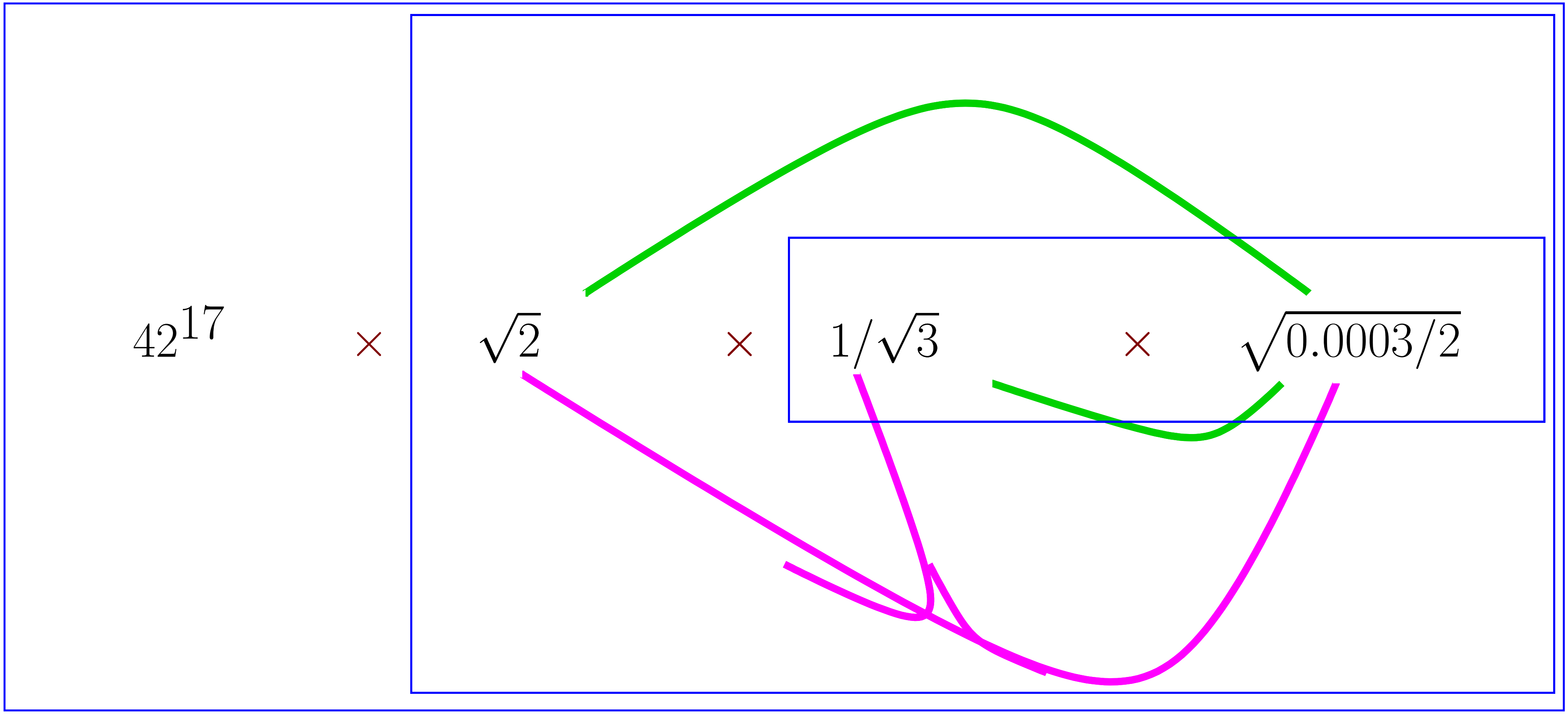
Outline

1. Hierarchical partitioning of sets with *structure*
2. Comparing partitionings in terms of *cost*
3. Algorithms for *incremental* partitioning
4. An application in model checking

Sets with Structure

$$42^{17} \quad \times \quad \sqrt{2} \quad \times \quad 1/\sqrt{3} \quad \times \quad \sqrt{0.0003/2}$$

Sets with Structure



structure ≙ hyperedges

good partitioning ≙ one where the edges cross few box borders

... are hypergraphs.

Hierarchical Partitioning + Cost

Given: Hypergraph $\mathcal{H} = (\mathcal{C}, \mathcal{E})$

Hierarchical Partitioning $\hat{=}$ tree $\mathcal{T}$ over leaves $\mathcal{C}$

boxes are internal nodes

root is the outermost box

For $e \in \mathcal{E}$: $\mathcal{T}_e :=$ smallest complete subtree *containing* e

$$\textit{depth\_cost}(\mathcal{T}) := \begin{cases} 2 & \text{if } \textit{depth}(\mathcal{T}) = 1 \\ \textit{depth}(\mathcal{T}) & \text{otherwise} \end{cases}$$

$$\textit{cost}(\mathcal{T}) := \sum_{e \in \mathcal{E}} \textit{depth\_cost}(\mathcal{T}_e) \cdot |\textit{leaves}(\mathcal{T}_e)|$$

NP-Optimization Problem

EDGE-GUIDED TREE-INDEXING
Given a hypergraph $\mathcal{H} = (\mathcal{C}, \mathcal{E})$ and a number $K \in \mathbb{N}$. Decide whether there exists a tree-indexing of cost at most K.

Fact:

EDGE-GUIDED TREE-INDEXING is NP-complete.

Reduction from MINIMUM CUT INTO EQUAL-SIZED SUBSETS [GJS76]**:**

Given: graph $G = (V, E)$

vertices $s, t \in V$, $K \in \mathbb{N}^{>0}$

Question: $\exists V_1 \uplus V_2 = V$. *(i)* $s \in V_1, t \in V_2$

(ii) $|V_1| = |V_2|$

(iii) $|E \cap V_1 \times V_2| \leq K$

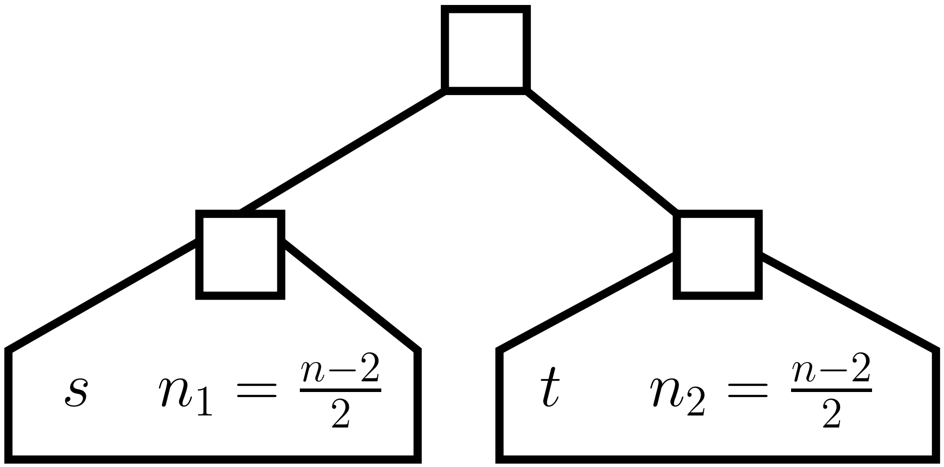
Sketch of Construction

$$(G, s, t, K) \quad \overset{f}{\mapsto} \quad (G',\ K')$$

$$\exists V_1, V2.\ \textit{(i)} \wedge \textit{(ii)} \wedge \textit{(iii)} \quad \Leftrightarrow \quad \exists \mathcal{T}_{G'}.\ \textit{cost}(\mathcal{T}_{G'}) \leq K'$$

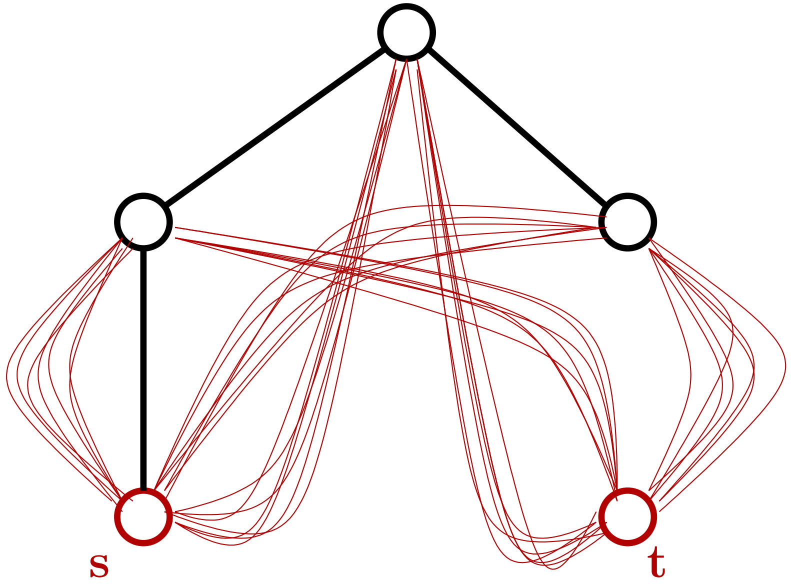
Idea: built G' such that a cost-optimal $\mathcal{T}_{G'}$ always looks like this:

$\mathcal{T}^*$:



Need: Upper bound $\boxed{\textit{cost}^* := \textbf{max}\ \ \textit{cost}\,(\mathcal{T}^*)}$

Attractors to s and t

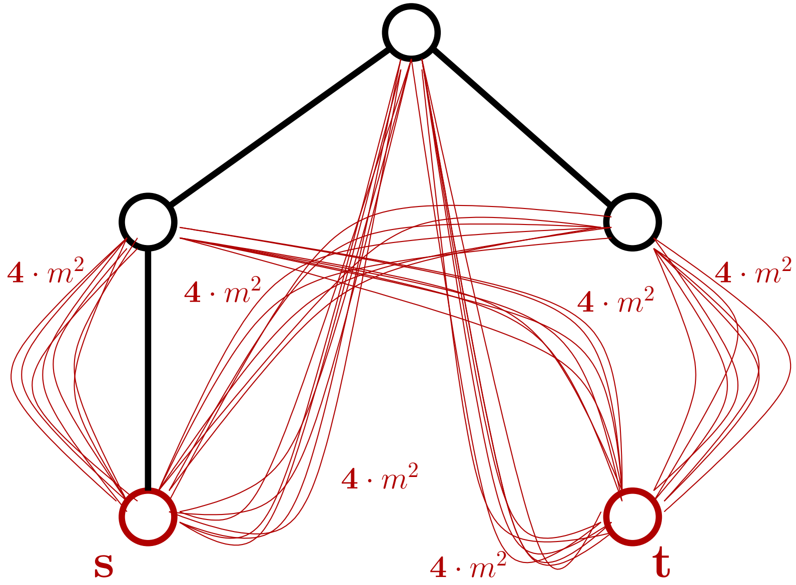


$G' := (V, E \cup \textit{Attractors})$

$$K' := \underline{\quad\quad} (2(\tfrac{n}{2}\text{-}1) \cdot 2n \cdot \tfrac{1}{2} + (n\text{-}2) \cdot 2n)$$
$$+ (m - K) \cdot 2n \cdot \tfrac{1}{2}$$
$$+ K \cdot 2n$$

$n := |V|, \ m := |E|$

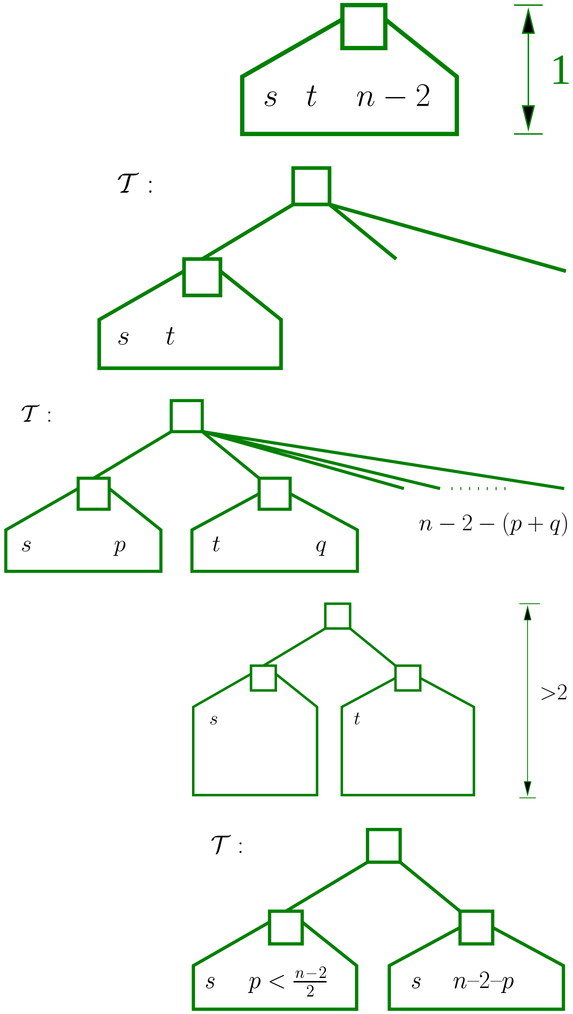
Attractors to s and t



$G' := (V, E \cup \textit{Attractors})$

$$K' := \underline{4m^2}(2(\tfrac{n}{2}\text{-}1)\cdot 2n \cdot \tfrac{1}{2} + (n\text{-}2)\cdot 2n)$$
$$+ (m-K)\cdot 2n \cdot \tfrac{1}{2}$$
$$+ K \cdot 2n$$

$n := |V|,\ m := |E|$



$\Rightarrow \textit{cost} > \textit{cost}^*$

$\Rightarrow \textit{cost} > \textit{cost}^*$

$\Rightarrow \textit{cost} > \textit{cost}^*$

$\Rightarrow \textit{cost} > \textit{cost}^*$

$\Rightarrow \textit{cost} > \textit{cost}^*$

Incremental Partitioning (heuristic)

input: hypergraph $\mathcal{H} = (\mathcal{C}, \mathcal{E})$

output: *forest* over leaves $\mathcal{C}$

PriorityQueue Q

forest $:= \mathcal{C}$

FORALL considered candidates $\mathcal{A} \subseteq \mathcal{C}$

    insert($\mathcal{A}$, Q)

WHILE *notempty*(Q)

    $\mathcal{A} :=$ *top*(Q)

    fresh node $\textcircled{\scriptsize A}$

    forest $:=$ *forest* $+ \ (\textcircled{\scriptsize A} \mapsto \mathcal{A})$

    FORALL $\mathcal{B} \in Q$ with $\mathcal{B} \cap \mathcal{A} \neq \varnothing$

        remove($\mathcal{B}$, Q)

    FORALL new candidates $\mathcal{D}$ containing $\textcircled{\scriptsize A}$

        insert($\mathcal{D}$, Q)

Parameters for the Algorithm

order of the priority queue

- given by a heuristic function $r_{\mathcal{E}} : 2^{\mathcal{C}} \longrightarrow \mathbb{R}$
- should favor small sub-forests that cover many hyperedges

selection of considered candidates

- restriction: consider candidates up to size k
- pre-computation: don't consider candidates that do not share a hyper-edge

Outline (revisited)

1. Hierarchical partitioning of *structured* sets ✔
2. Comparing partitionings in terms of *cost* ✔
3. Algorithms for *incremental* partitioning ✔
4. An application in model checking

Model Checking

$$\boxed{M \overset{?}{\models} \varphi}$$

M	:	description of the system
φ	:	desired property

- easier than proving a general theorem
- completely automatic ('yes' or counterexample)
- *efficient* algorithms tailored for classes of problems

Model Checking with MOCHA

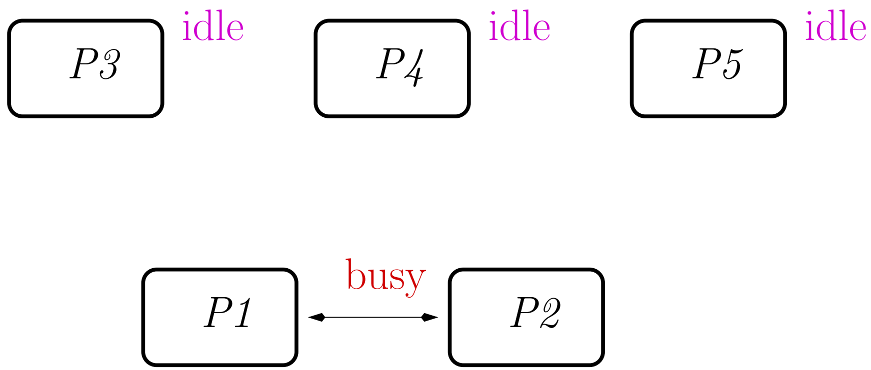
Spec: Reactive Modules

- parallel execution of components
- round-based or completely asynchronous
- communication via shared variables (1 write/multi read)

Logic: ATL (branching time)

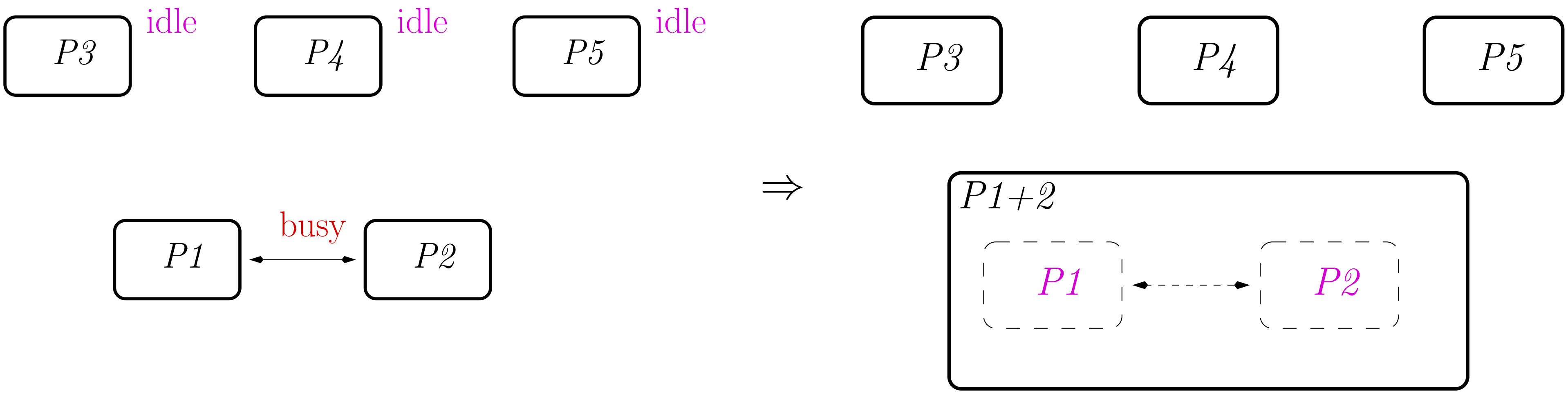
- CTL $\subset$ *ATL* $\subset$ μ-calculus
- notion of *strategy* to reach a goal
- invariant check:
  allows temporal scaling via "next" Θ for P
  (*Rajeev Alur and Bow-Yaw Wang, CONCUR'99*)
- heuristic to preprocess a system for "next"

Temporal Scaling



$$P1 \,||\, P2 \,||\, P3 \,||\, P4 \,||\, P5$$

Temporal Scaling



Instead of:

$$P1 \,||\, P2 \,||\, P3 \,||\, P4 \,||\, P5$$

Use:

$$\textit{hide}\ busy\ \textit{in}\ \boxed{P1 \,||\, P2} \,||\, P3 \,||\, P4 \,||\, P5$$

The *“Next”* Heuristic

Given : system S of reactive modules
structure and variables to hide
sub-system P
set Θ of transitions entering/leaving P

Computes : reactive module expression *next* Θ *for* P

Fact : for reachability analysis,
we can replace P by *next* Θ *for* P
without changing the answer

next Θ *for* P is ignoring irrelevant behavior; this gives a speedup.

Problem: *"Next"* Requires Preprocessing

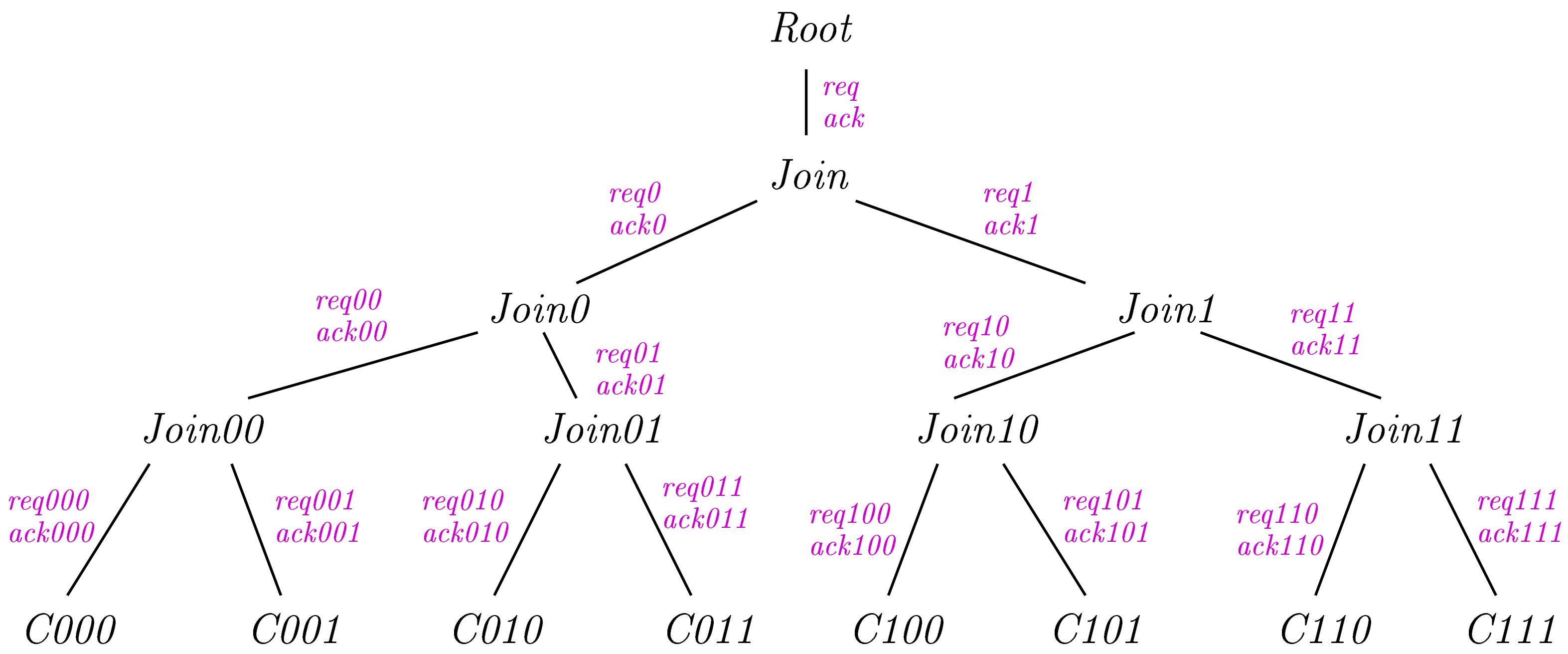
We cannot apply 'next' on ¤at structures:

We have to tell, *where* to hide and *what*.

```
module Sys is    Root
              || (hide req0,ack0,req1,ack1 in
                  (   Join
                   || (hide req00,ack00,req01,ack01 in (J0 || C00 || C01))
                   || C1))
```

- tedious to do by hand
- not always obvious what is a *good* structure

Asynchronous Parity Computer

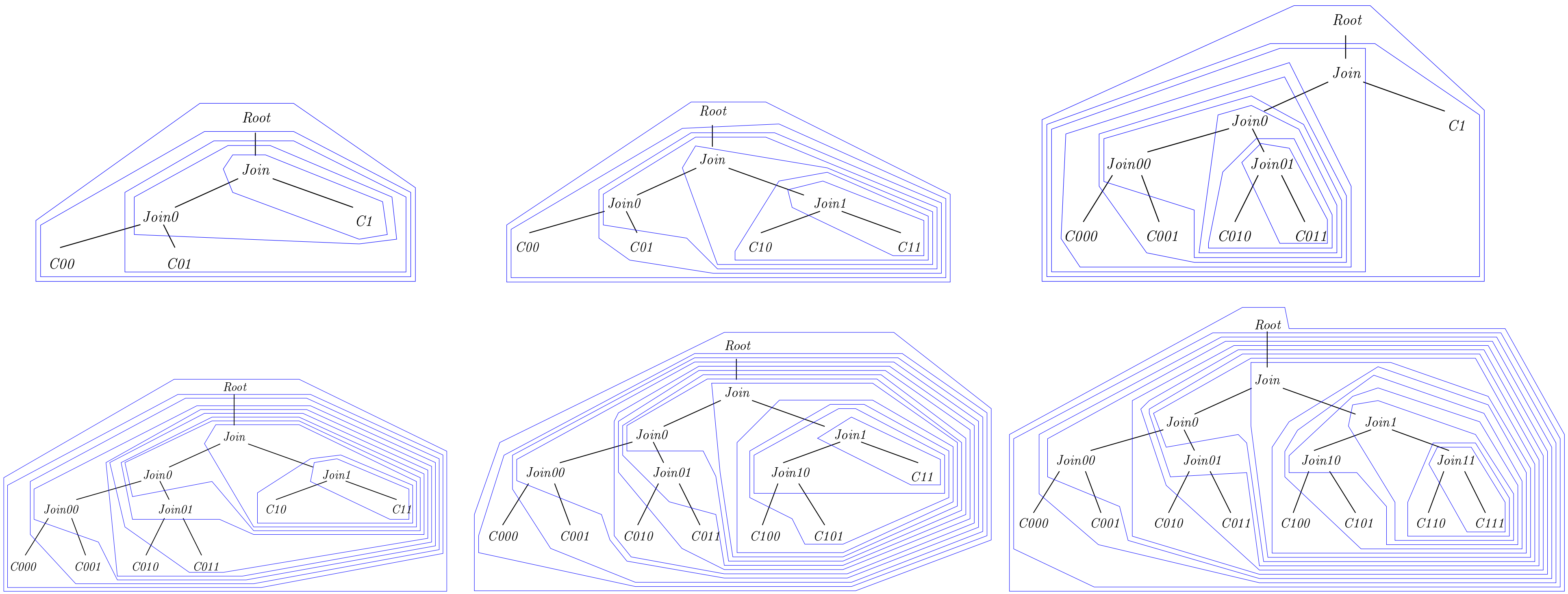


Clients : issue value *true* or *false*

Joins : compute xor

Root : acknowledges

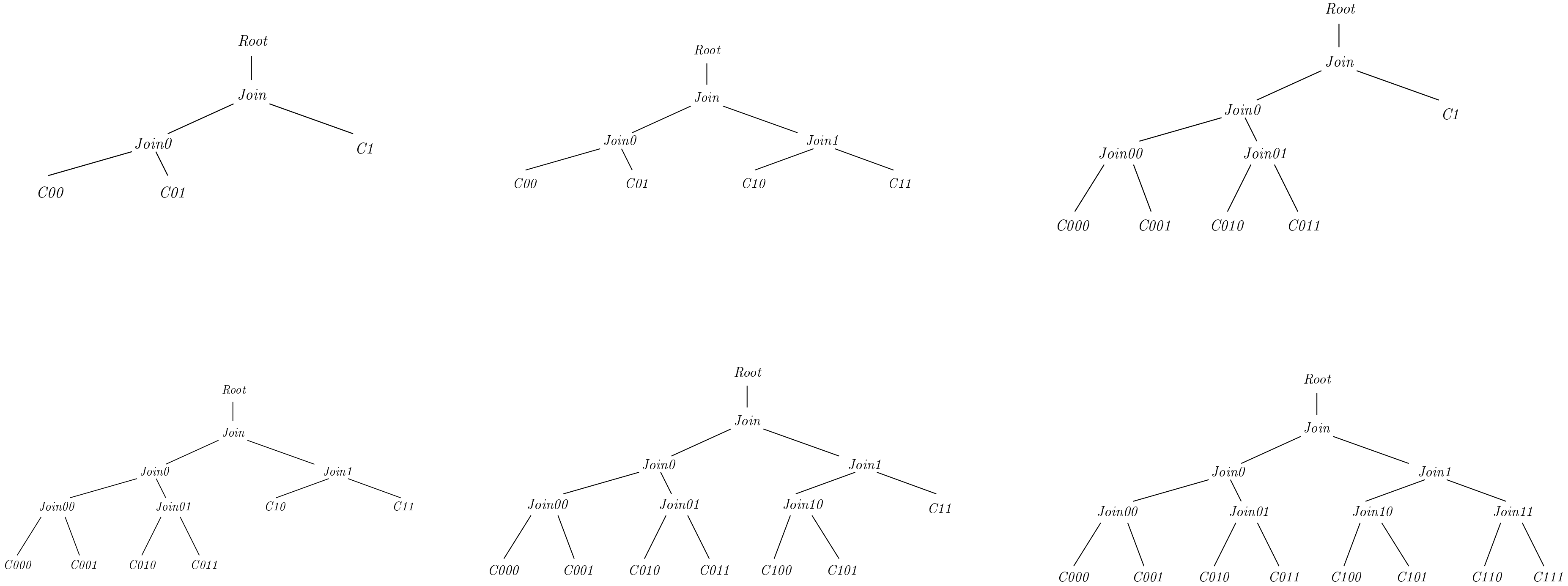
Bad Heuristic Function



$$\mathbf{r}_{pref}(\mathcal{A}) := \frac{\left|\{e \in \mathcal{E} \,\middle|\, e \subseteq \mathcal{A}\}\right|}{|\mathcal{A}|^2}$$

$\Rightarrow$ Ignores *depth* of a sub-structure

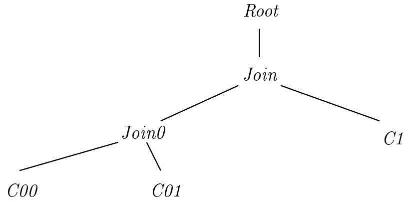
Good Heuristic Function

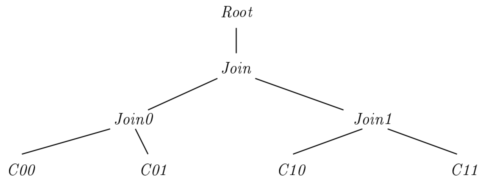


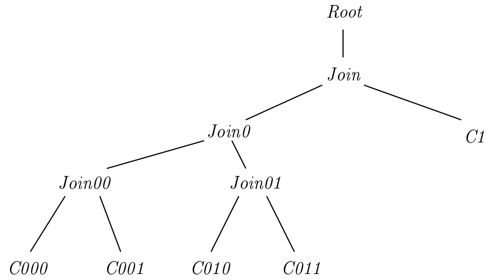
$$\mathbf{r}_{pref} \quad (\mathcal{A}) := \frac{\left|\left\{ e \in \mathcal{E} \;\middle|\; e \subseteq \mathcal{A} \right\}\right|}{|\mathcal{A}|^2}$$

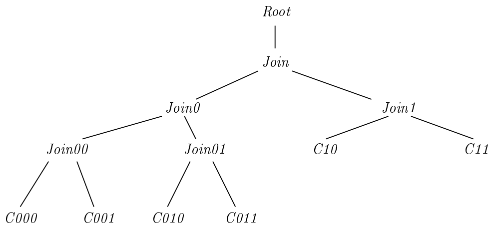
Cover-Number　　　Size

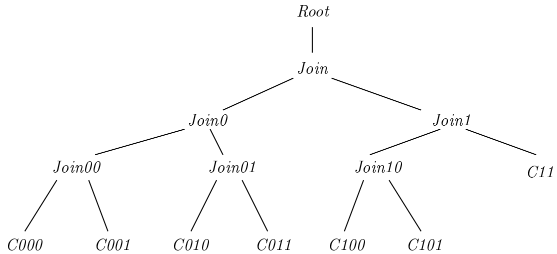
Good Heuristic Function

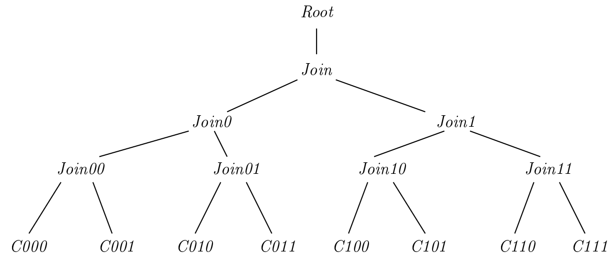








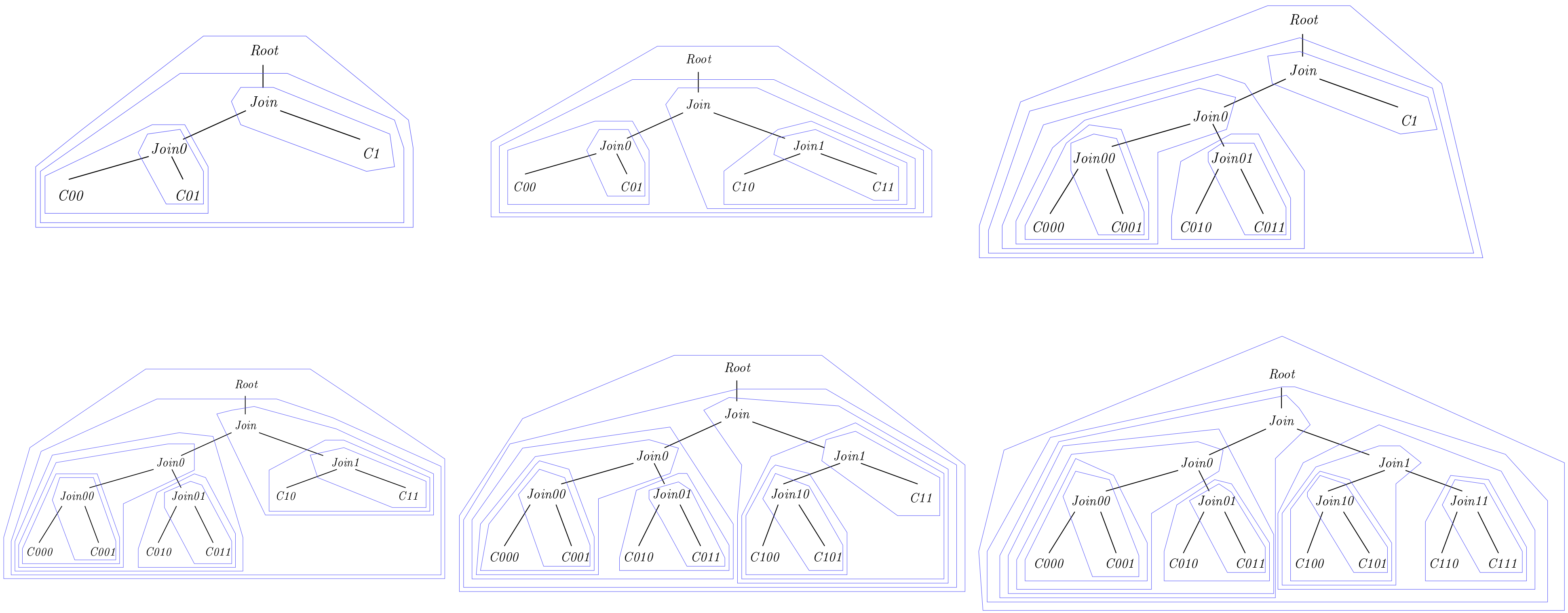




$$\mathbf{r}_{pref}{}^{+}(\mathcal{A}) := \frac{|\{e \in \mathcal{E} \mid e \subseteq \mathcal{A}\}|}{|\mathcal{A}|^2} + \frac{\varepsilon_1}{|\{e \in \mathcal{E} \mid e \cap \mathcal{A} \neq \emptyset\}|} + \frac{\varepsilon_2}{\textit{depth}(\mathcal{A})}$$

Cover-Number Size Egdes Depth

Good Heuristic Function



$$\mathbf{r}_{pref}{}^{+}(\mathcal{A}) := \frac{\left|\{e \in \mathcal{E} \mid e \subseteq \mathcal{A}\}\right|}{|\mathcal{A}|^2} + \frac{\varepsilon_1}{\left|\{e \in \mathcal{E} \mid e \cap \mathcal{A} \neq \emptyset\}\right|} + \frac{\varepsilon_2}{\mathit{depth}(\mathcal{A})}$$

Cover-Number Size Egdes Depth

Parity Computer: Runtime Comparison

N	partition	$\|$hash$\|$	check
3	3˙227	97	556
4	4˙683	647	3˙507
5	6˙214	1˙945	11˙442
6	9˙314	16˙047	102˙920
7	19˙064	58˙353	433˙828
8	69˙006	*o.o.Mem*	–

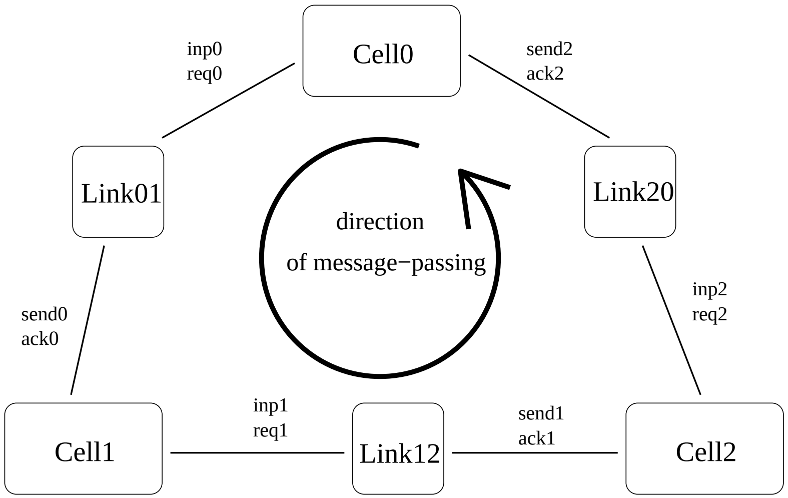
Applying $\mathbf{r}_{pref}$ as heuristic function

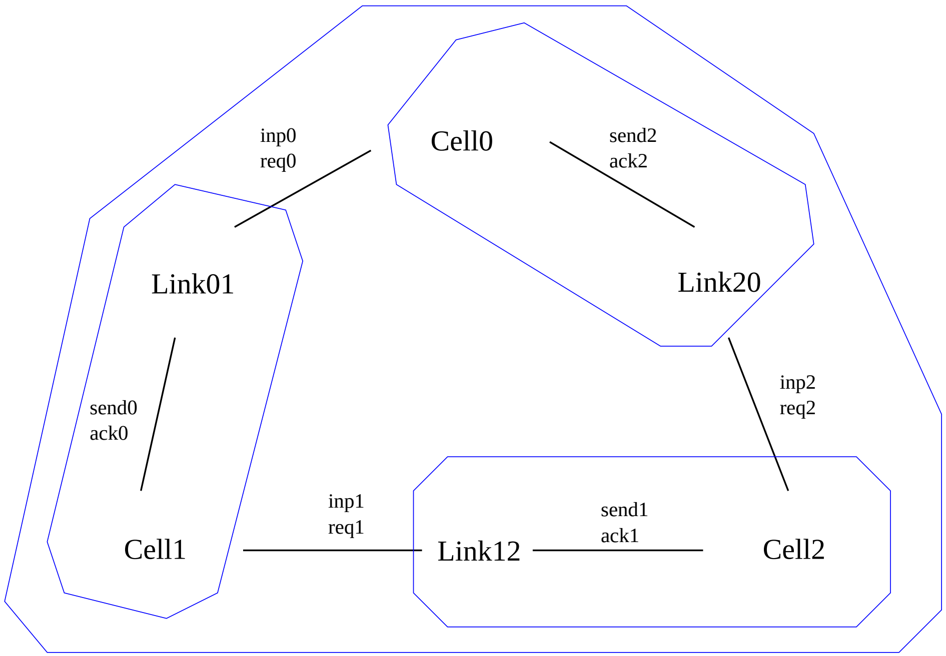
N	partition	$\|$hash$\|$	check
3	134	53	349
4	313	119	787
5	712	141	1˙146
6	2˙742	207	1˙813
7	12˙804	273	2˙632
8	63˙834	471	4˙973

Applying $\mathbf{r}^{+}_{pref}$ with

$$\varepsilon_1 := \frac{1}{1000},\ \varepsilon_2 := \frac{1}{100000}$$

Leader Election Protocol





size	|hash|	check
2	563	6'270
3	70'797	1'327'756

size	partition	|hash|	check
2	217	563	4'615
3	279	61'455	661'275

Summary

We established

- ★ incremental method for hierarchical partitioning
  (and implemented it in a model checking tool)
- ★ heuristic function based on 4 criteria:
  Cover-Number, Size, Edges, Depth
- ★ sample problems, where this heuristic is well-behaved

We don't know (yet)

- ☆ the computational complexity is for other *cost* functions
- ☆ whether there exist *polynomial approximation schemes*
- ☆ how to *exploit* hierarchical partitionings by other means
  (e.g., abstractions)

References

[AW99] Rajeev Alur and Bow-Yaw Wang. “Next” Heuristic for On-the-¤y Model Checking. In *Proceedings of the Tenth International Conference on Concurrency Theory (CONCUR’99)*, LNCS 1664, pages 98–113. Springer-Verlag, 1999.

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simpli£ed NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, February 1976.

[jmo00] Mocha: Exploiting Modularity in Model Checking, 2000. see `http://www.cis.upenn.edu/~mocha`.

[MA00] M. Oliver Möller and Rajeev Alur. Heuristics for hierarchical partitioning with application to model checking. Research Series RS-00-21, BRICS, Department of Computer Science, University of Aarhus, August 2000. 30 pp, available online at `http://www.brics.dk/RS/00/21/`.

[Sch70] Ernst Schröder. Vier combinatorische probleme. *Zentralblatt. f. Math. Phys.*, 15:361–376, 1870.