Flattening Statecharts without Explosions

Andrzej Wąsowski

Department of Innovation
IT University of Copenhagen

http://www.mini.pw.edu.pl/˜wasowski/scope/

LCTES 2004, Washington, DC, June 13, 2004

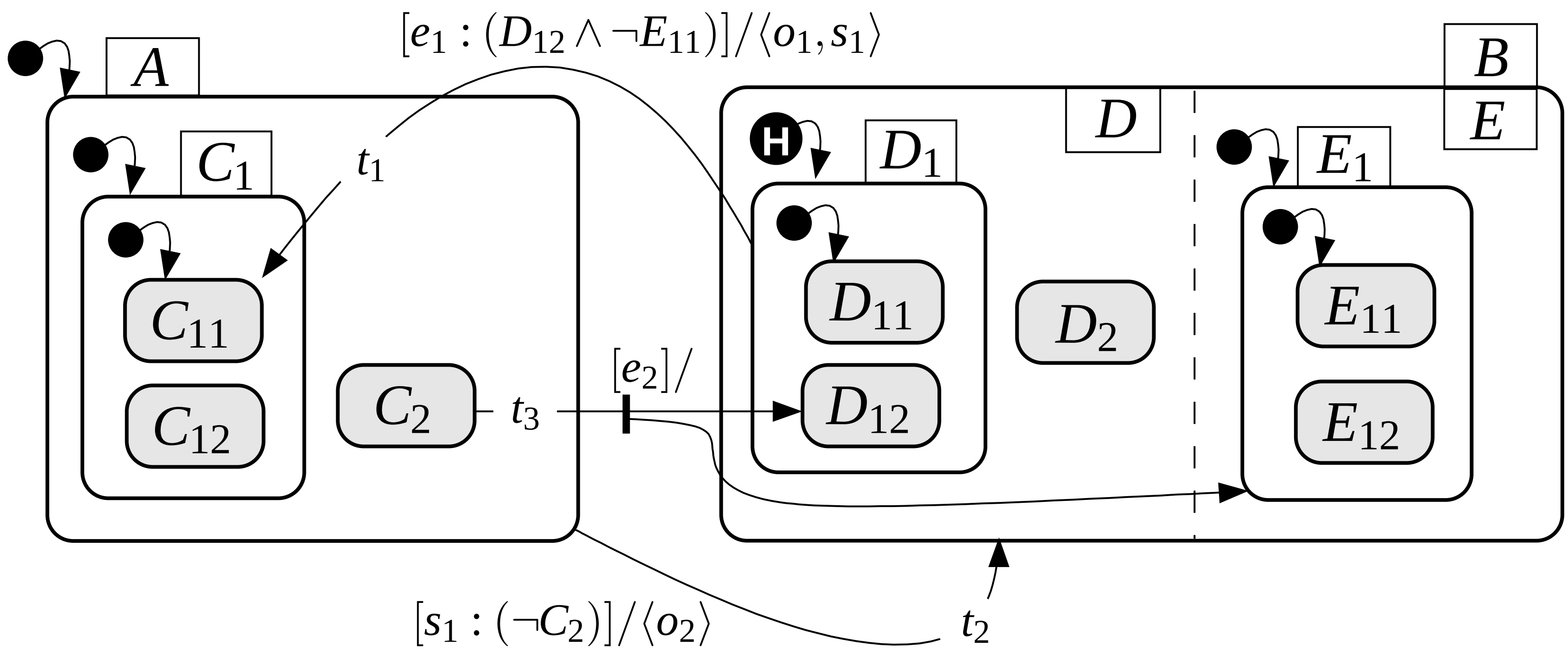
Outline

- **Introduction and Motivation**
  - Hierarchical Statecharts
  - Flat Statecharts
  - The Flattening problem
- **A Lower Bound for Flattening**
- **Polynomial flattening for Code Generation**
- **Experimental results**
- **Conclusion**

Outline

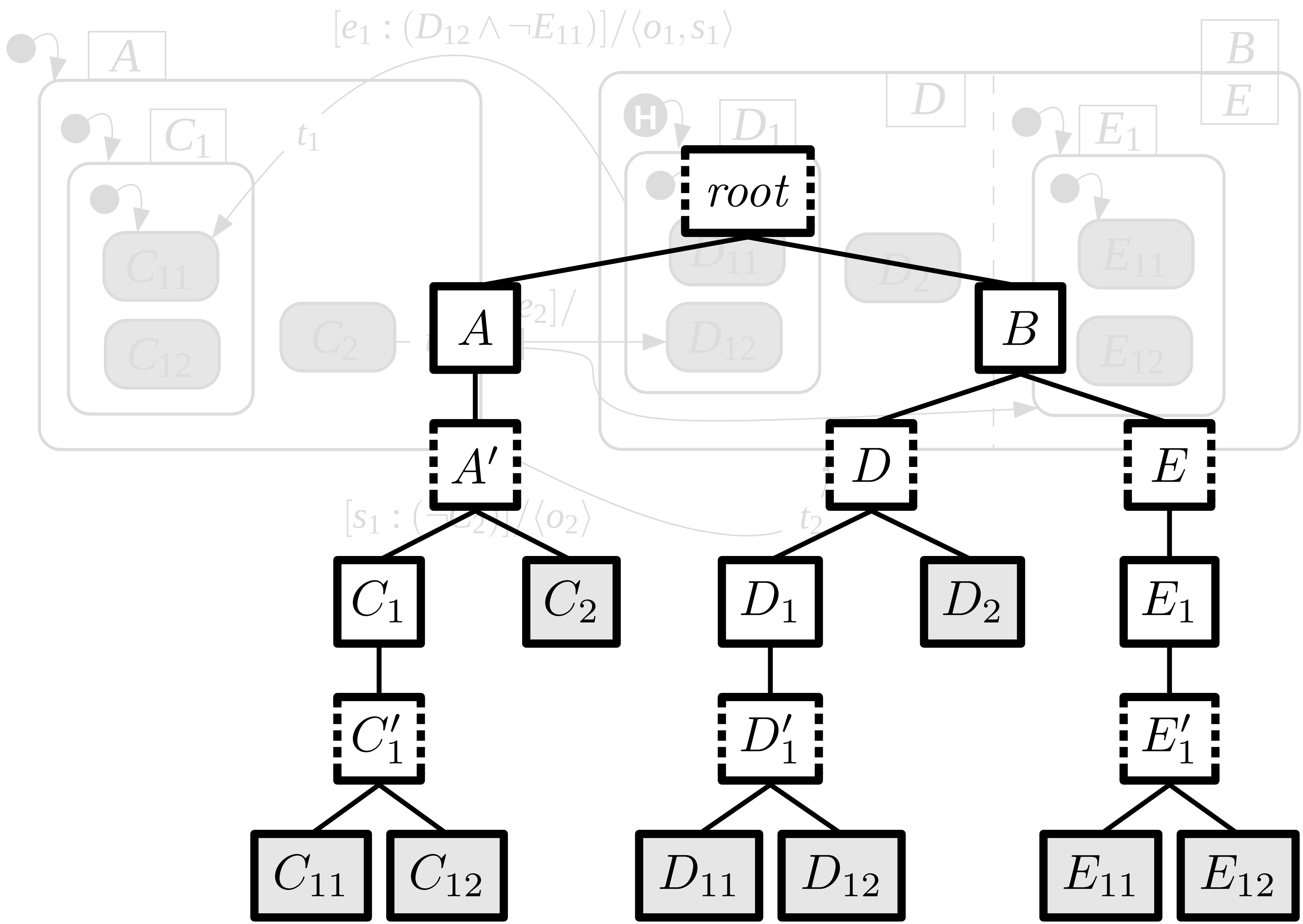
- **Introduction and Motivation**
  - Hierarchical Statecharts
  - Flat Statecharts
  - The Flattening problem
- **A Lower Bound for Flattening**
- **Polynomial flattening for Code Generation**
- **Experimental results**
- **Conclusion**

Hierarchical Statecharts

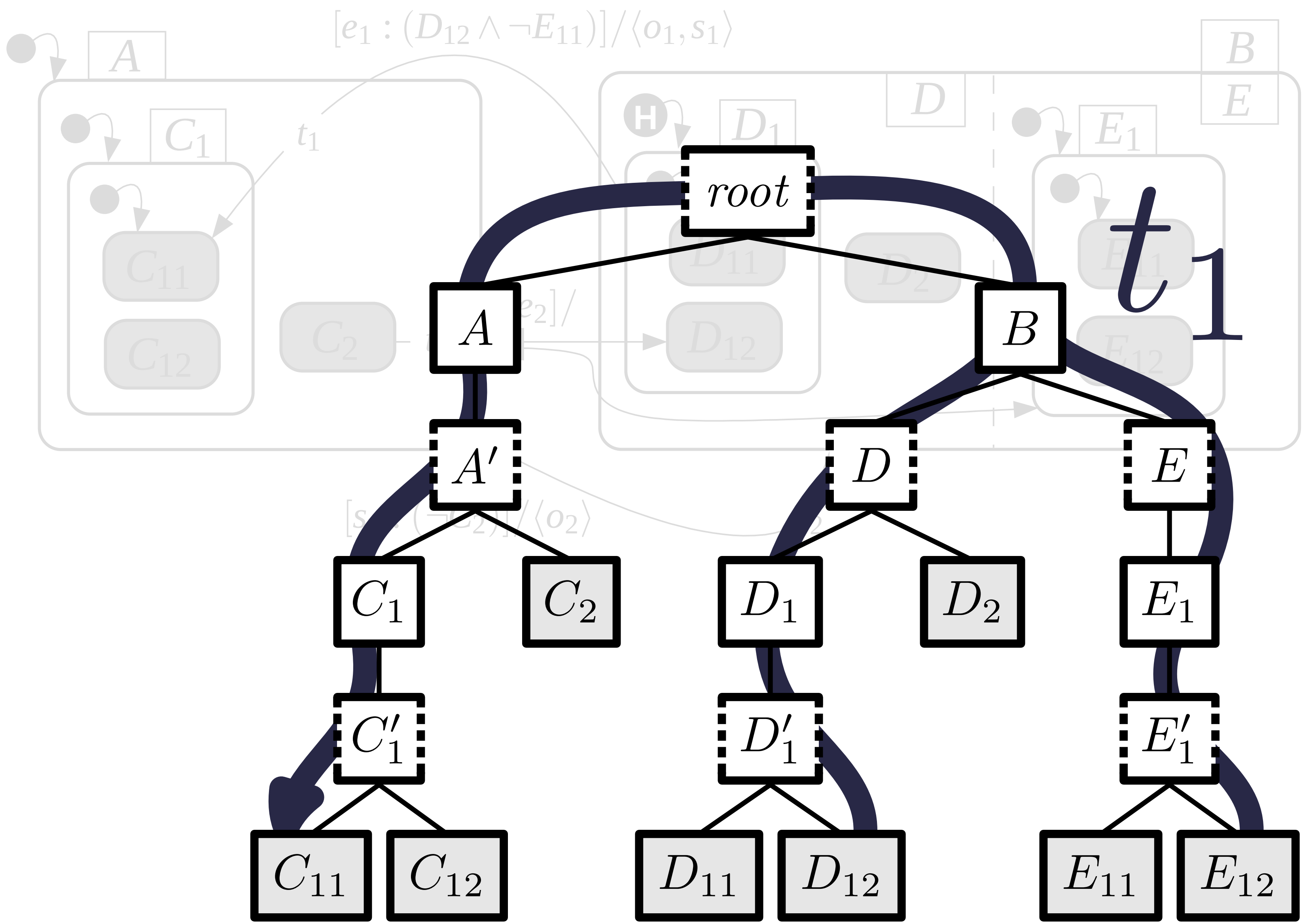


- Entry & exit actions on every state.
- Transitions guarded by events and current configuration.
- Signals (local events), signal queue.
- Concurrent, but usually compiled to sequential code.

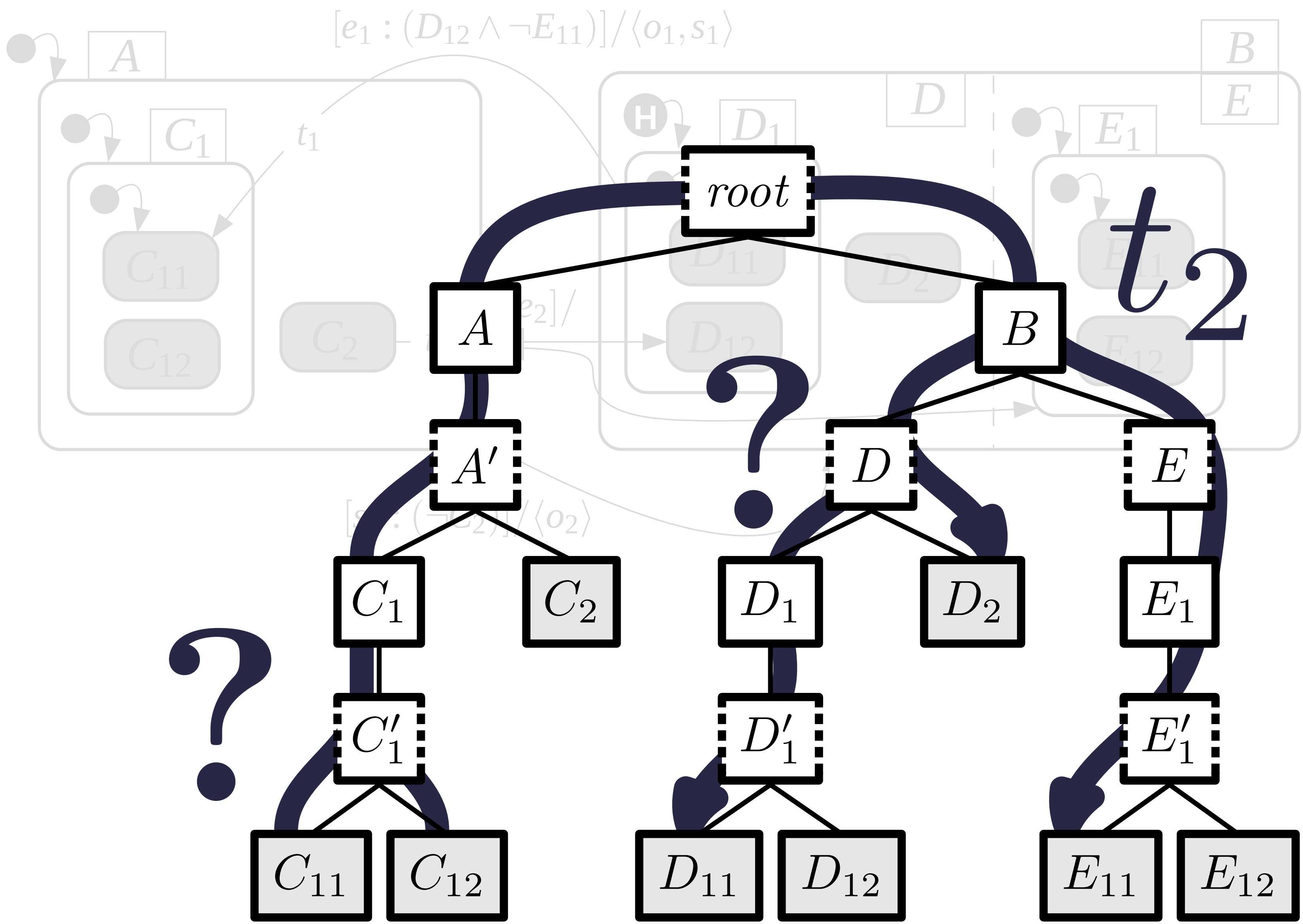
Hierarchical Statecharts (II)



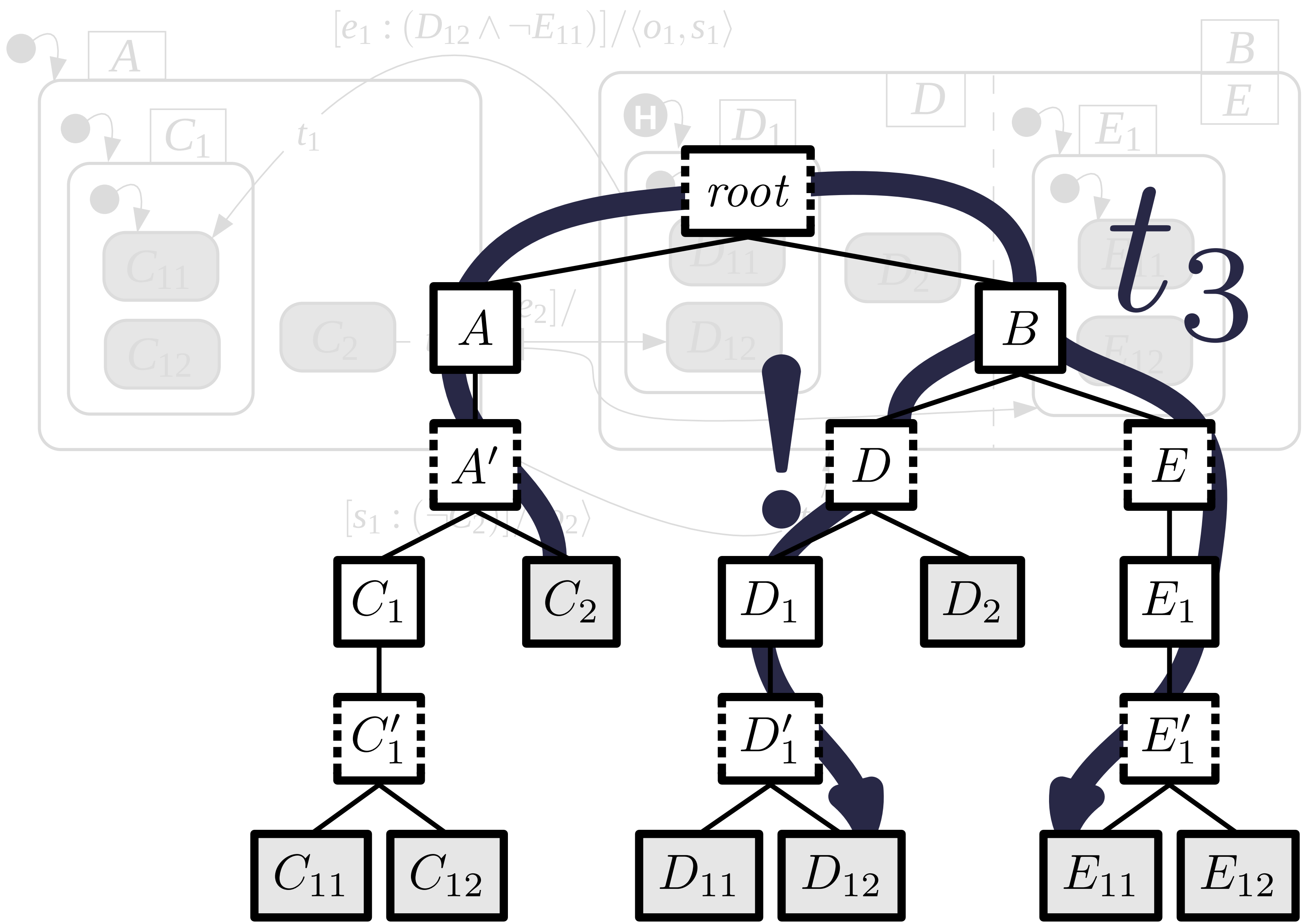
Hierarchical Statecharts (III)



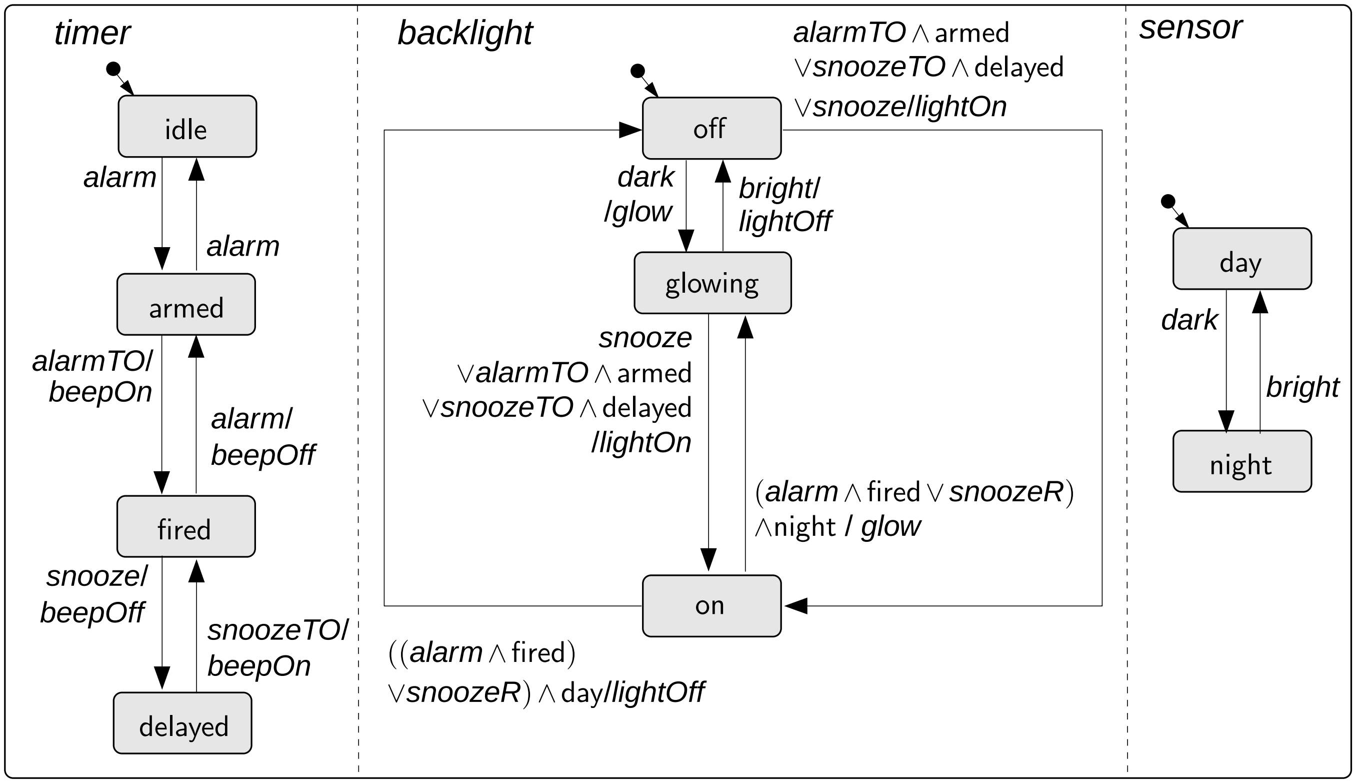
Hierarchical Statecharts (IV)



Hierarchical Statecharts (V)



Flat Statecharts

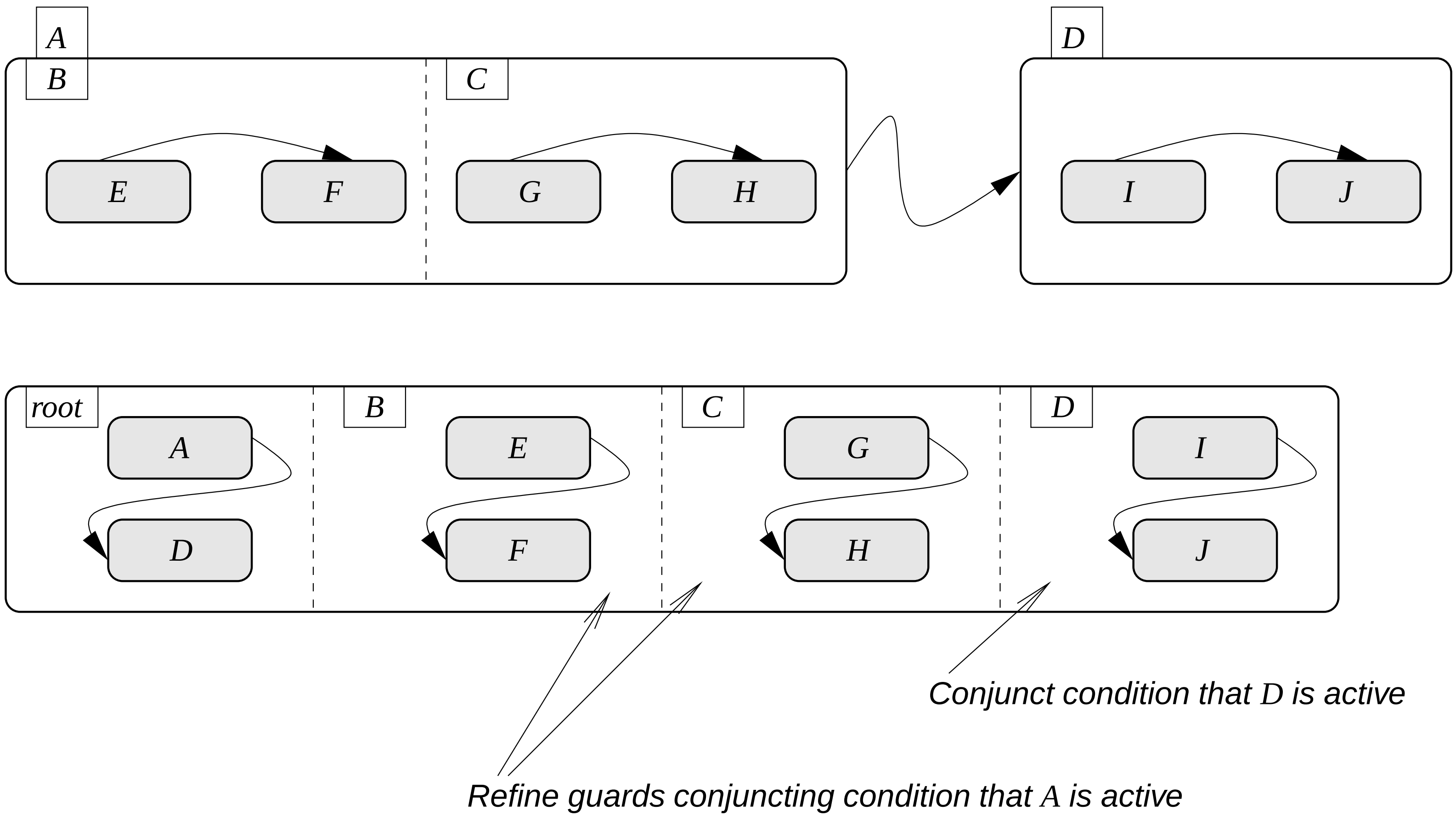


- Set of Mealy Machines
- No entry/exit actions in states
- Synchronization on states
- Configurations = state vectors
- All machines active all the time
- Hier. statechart with history
- Flat transition scopes
- Nondeterministic order

Flat Statecharts (II)

$$
\begin{array}{rl}
\texttt{case}\ e\ \texttt{of} & \\
\mathit{alarm}: & \textbf{if}\ \mathsf{idle}\ \textbf{then}\ \sigma'[0] := \mathsf{armed}; \\
& \textbf{if}\ \mathsf{idle}\ \textbf{then}\ \sigma'[0] := \mathsf{idle}; \\
& \textbf{if}\ \mathsf{fired}\ \textbf{then}\ \sigma'[0] := \mathsf{armed}; \mathit{beepOff}; \\
& \textbf{if}\ \mathsf{fired} \wedge \mathsf{night} \wedge \mathsf{on}\ \textbf{then}\ \sigma'[1] := \mathsf{glowing}; \mathit{glow}; \\
& \textbf{if}\ \mathsf{fired} \wedge \mathsf{day} \wedge \mathsf{on}\ \textbf{then}\ \sigma'[1] := \mathsf{off}; \mathit{lightOff}; \\
\mathit{alarmTO}: & \textbf{if}\ \mathsf{armed}\ \textbf{then}\ \sigma'[0] := \mathsf{fired}; \mathit{beepOn}; \\
& \textbf{if}\ \mathsf{armed} \wedge \mathsf{off}\ \textbf{then}\ \sigma'[1] := \mathsf{on}; \mathit{lightOn}; \\
& \textbf{if}\ \mathsf{armed} \wedge \mathsf{glowing}\ \textbf{then}\ \sigma'[1] := \mathsf{on}; \mathit{lightOn}; \\
\mathit{snooze}: & \textbf{if}\ \mathsf{fired}\ \textbf{then}\ \sigma'[0] := \mathsf{delayed}; \mathit{beepOff} \ldots \\
\mathit{snoozeTO}: & \textbf{if}\ \mathsf{delayed}\ \textbf{then}\ \sigma'[0] := \mathsf{fired}; \mathit{beepOn} \ldots \\
\mathit{dark}: & \textbf{if}\ \mathsf{off}\ \textbf{then}\ \sigma'[1] := \mathit{glowing}; \mathit{glow} \ldots \\
\mathit{bright}: & \textbf{if}\ \mathsf{glowing}\ \textbf{then}\ \sigma'[1] := \mathit{off}; \mathit{lightOff} \ldots
\end{array}
$$

Flattening = Hierarchy Elimination



- Unfortunately becomes complex in presence of exit actions.
- What are the exit actions executed by top transition?

Flattening (II)

Semantics: an I/O Alternating Labeled Transition System.

[Simulation] Statechart configuration σ_0 simulates a state configuration σ'_0, written $\sigma'_0 \leqslant \sigma_0$, iff

whenever $\sigma'_0 \xrightarrow{e \quad os} \sigma'_1$

then also $\sigma_0 \xrightarrow{e \quad os} \sigma_1$ and $\sigma'_1 \leqslant \sigma_1$.

Note! input-enabled setting and non-determinism.

[Flattening] Let F be an algorithm transforming statecharts. F is a flattening algorithm if for any hierarchical S it yields a flat S' such that $S' \lesssim S$.

Outline

- **Introduction and Motivation**
  - Hierarchical Statecharts
  - Flat Statecharts
  - The Flattening problem
- **A Lower Bound for Flattening**
- **Polynomial flattening for Code Generation**
- **Experimental results**
- **Conclusion**

A Lower Bound on Flattening

[Theorem]
There exists a hierarchical statechart S such that for any flat statechart S' implementing it, $S' \lesssim S$, such that S' does not use signals: <u>the size of S' is in $\Omega(2^{\sqrt{s}})$</u>, where s represents the size of S.

[Wasowski,SFEDL'04]

Outline

- **Introduction and Motivation**
  - Hierarchical Statecharts
  - Flat Statecharts
  - The Flattening problem
- **A Lower Bound for Flattening**
- **Polynomial flattening for Code Generation**
- **Experimental results**
- **Conclusion**

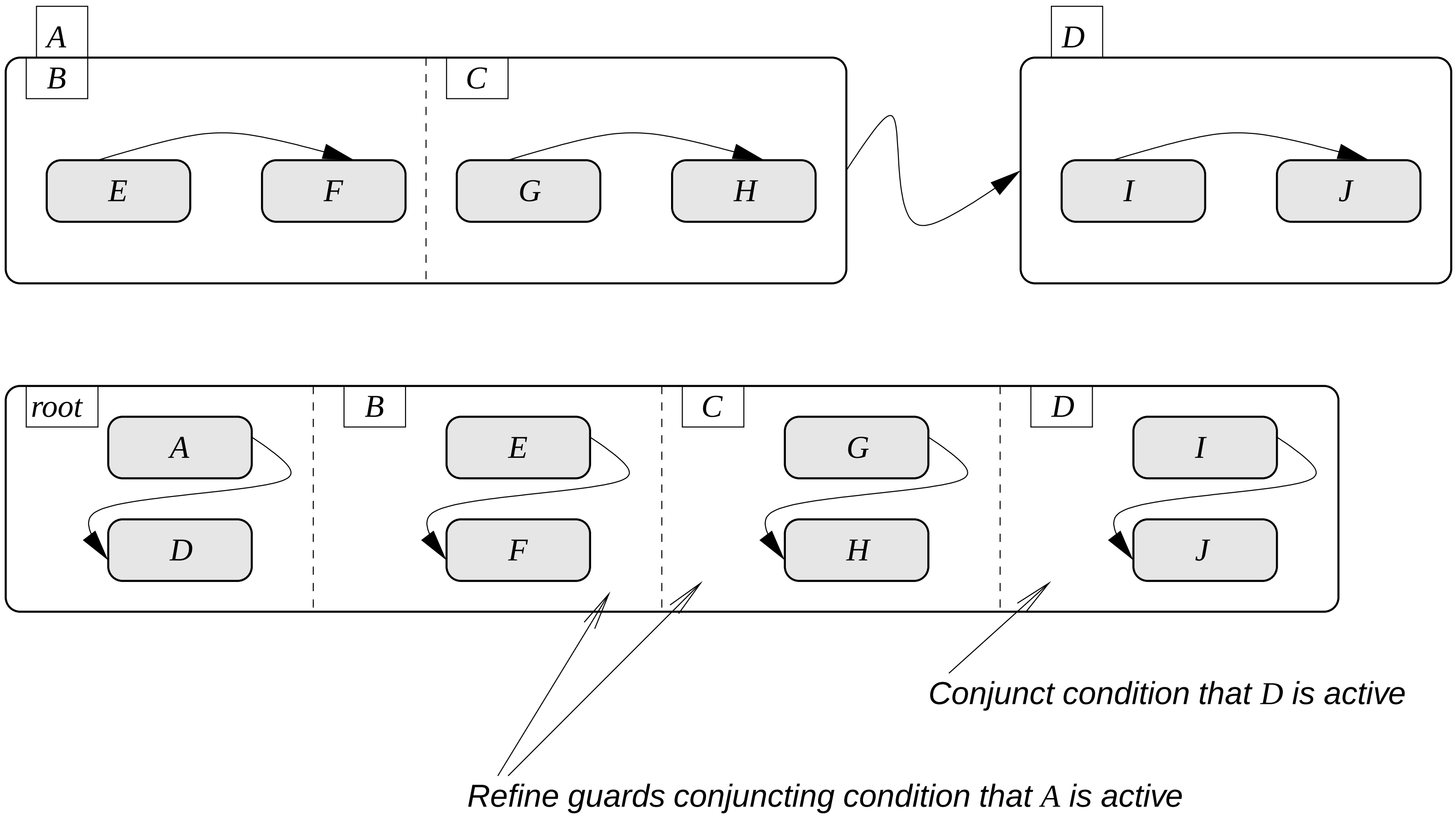
Non-explosive Flattening

[Theorem] For any hierarchical UML statechart S there exists a flat statechart S' with queue-based signal communication such that $S' \lesssim S$ and the size of S' is at most polynomial in the size of S.

A signal queue in the output language, or the sequence of rule execution, can be used to achieve flattening in polynomial space.

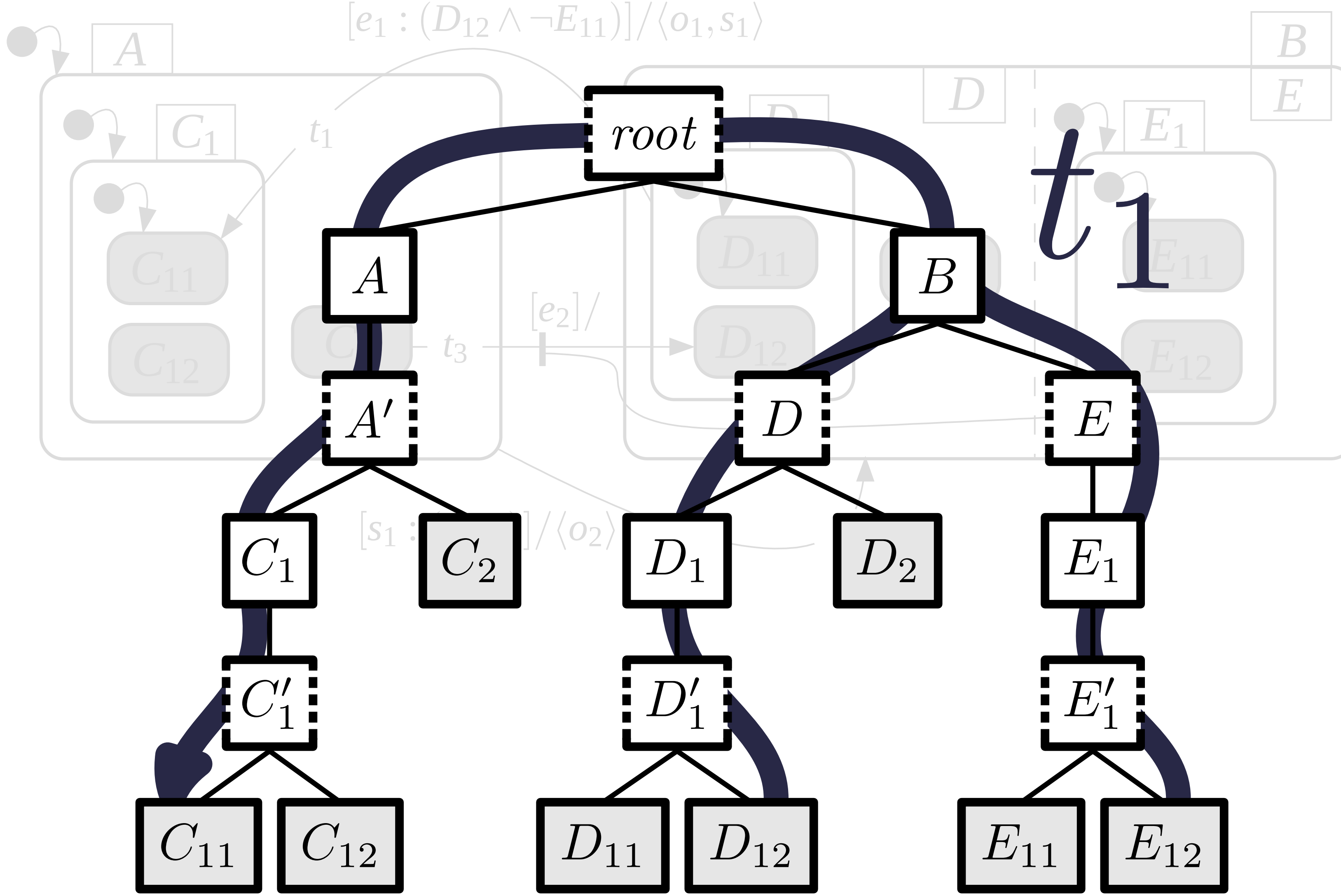
This is the approach of SCOPE code generator.

Flatten the state tree first

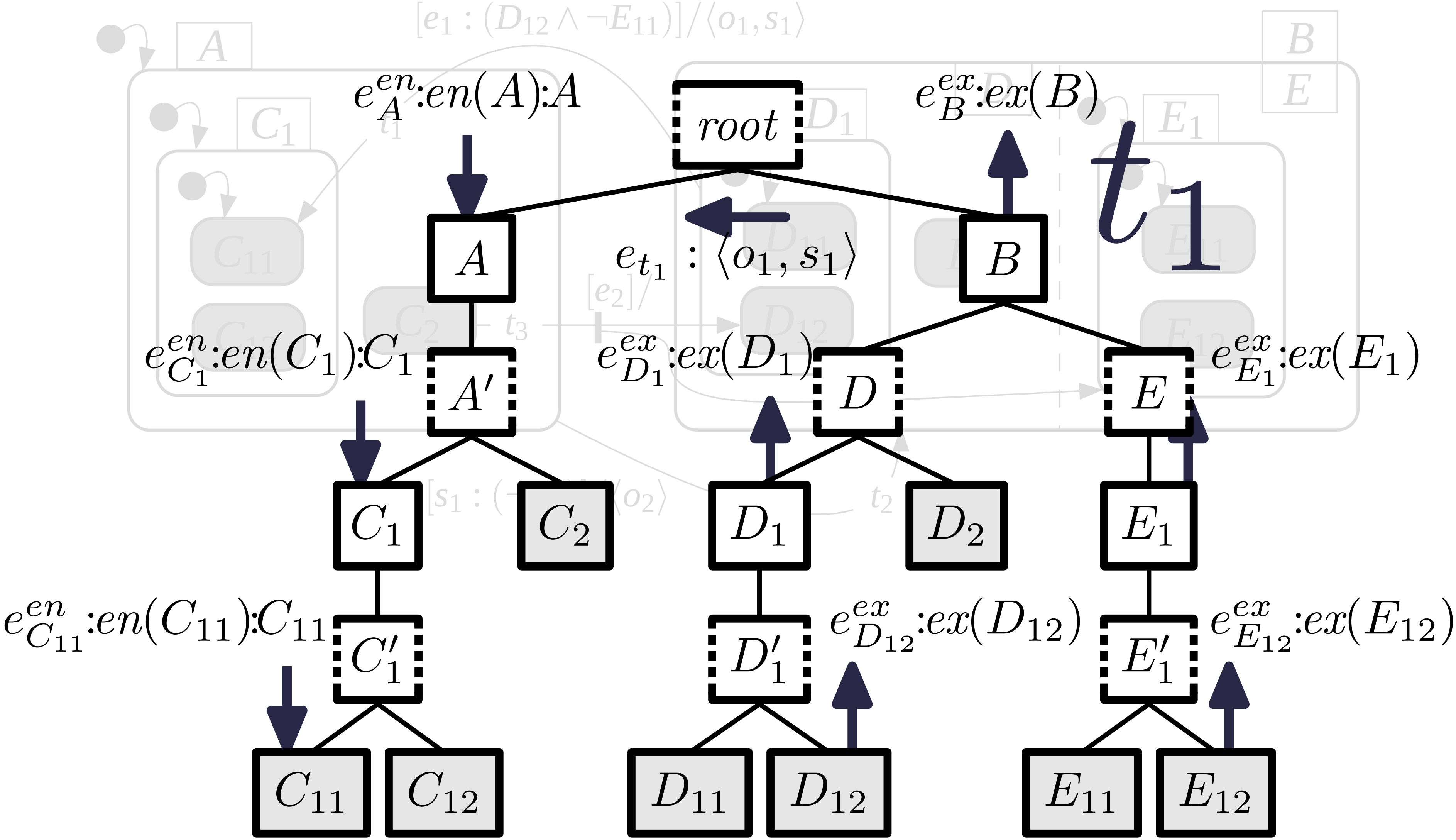


- Hierarchy represented explicitly in guards: substitute each occurence of state s with a conjunction of s and its ancestors.
- Clearly no explosion here.

Flatten hierarchical transitions

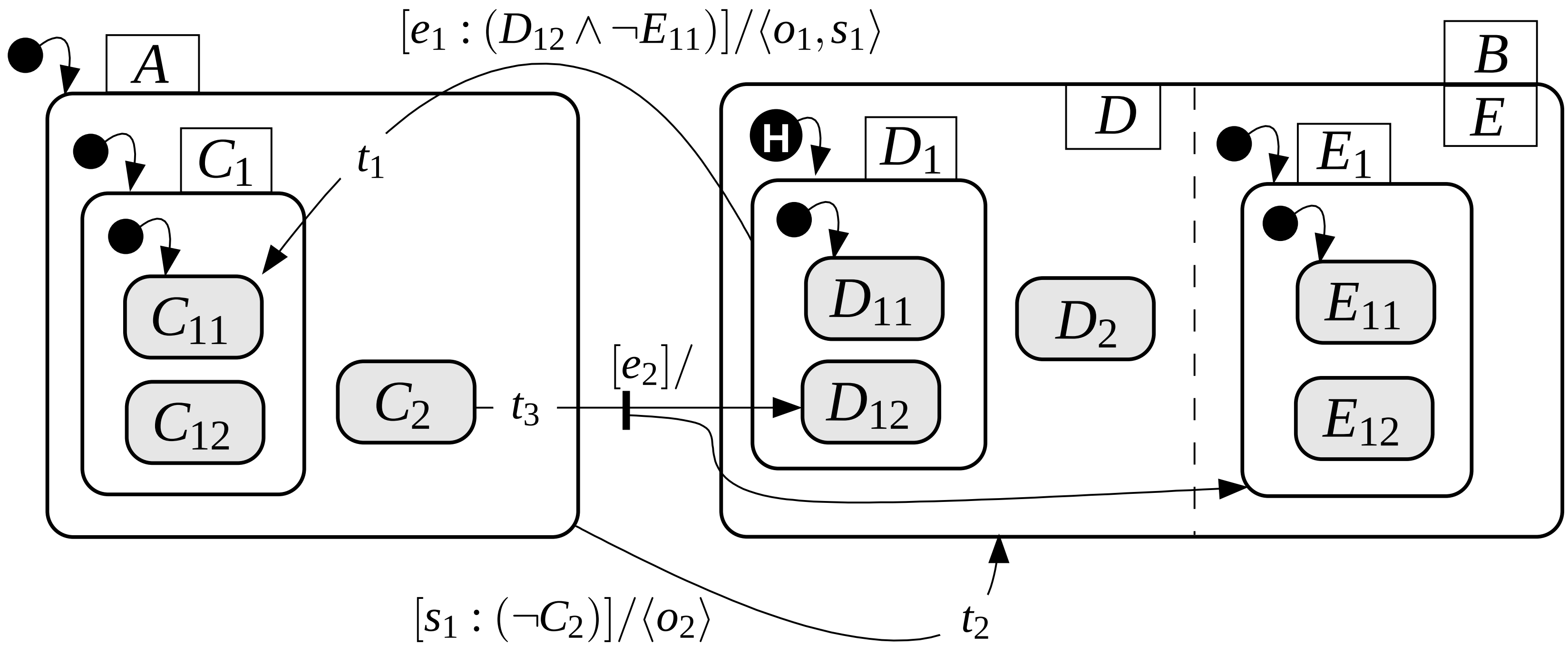


Flatten hierarchical transitions (II)



At most as many new transitions as the number of states.

Flatten hierarchical transitions (III)

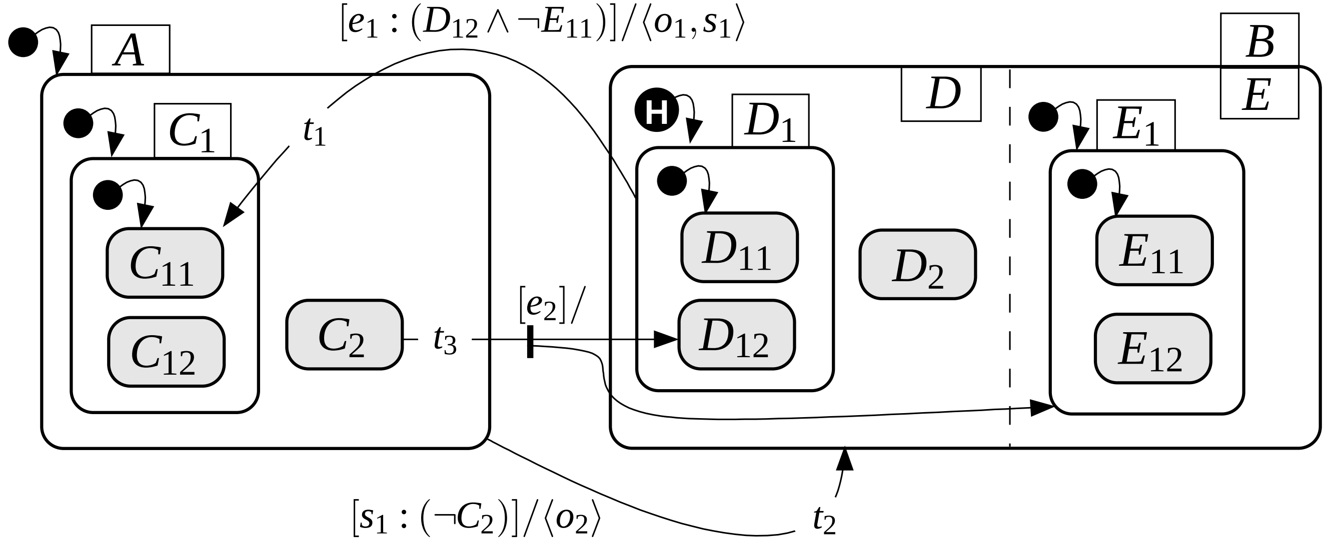


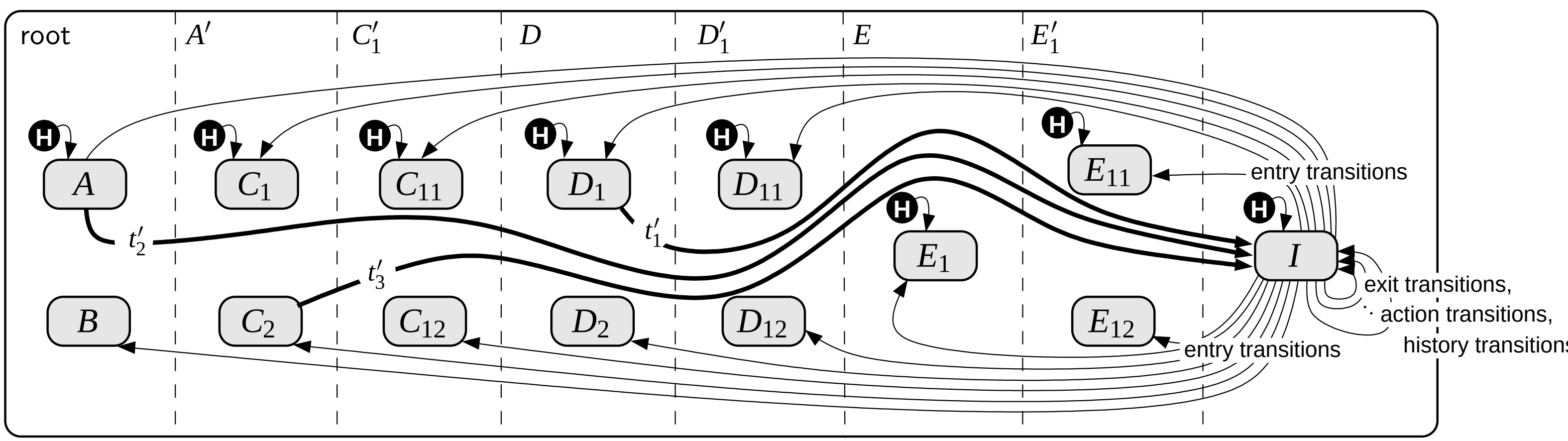
- Add a scheduling transition:

$$t_1' = D_1 \xrightarrow{[e_1:(D_1\wedge B)\wedge((D_{12}\wedge D_1\wedge B)\wedge\neg(E_{11}\wedge E_1\wedge B))]/\langle e^{\mathrm{ex}}_{D_{12}}, e^{\mathrm{ex}}_{D_1}, e^{\mathrm{ex}}_{E_{11}}, e^{\mathrm{ex}}_{E_{12}}, e^{\mathrm{ex}}_{E_1}, e^{\mathrm{ex}}_{B}, e_{t_1}, e^{\mathrm{en}}_{A}, e^{\mathrm{en}}_{C_1}, e^{\mathrm{en}}_{C_{11}}\rangle} I$$

- Note that this only works due to weak conformance requirement (simulation). Only one possible sequence is chosen.
- In simple-minded translation more exit signals are added.

What comes out of it?

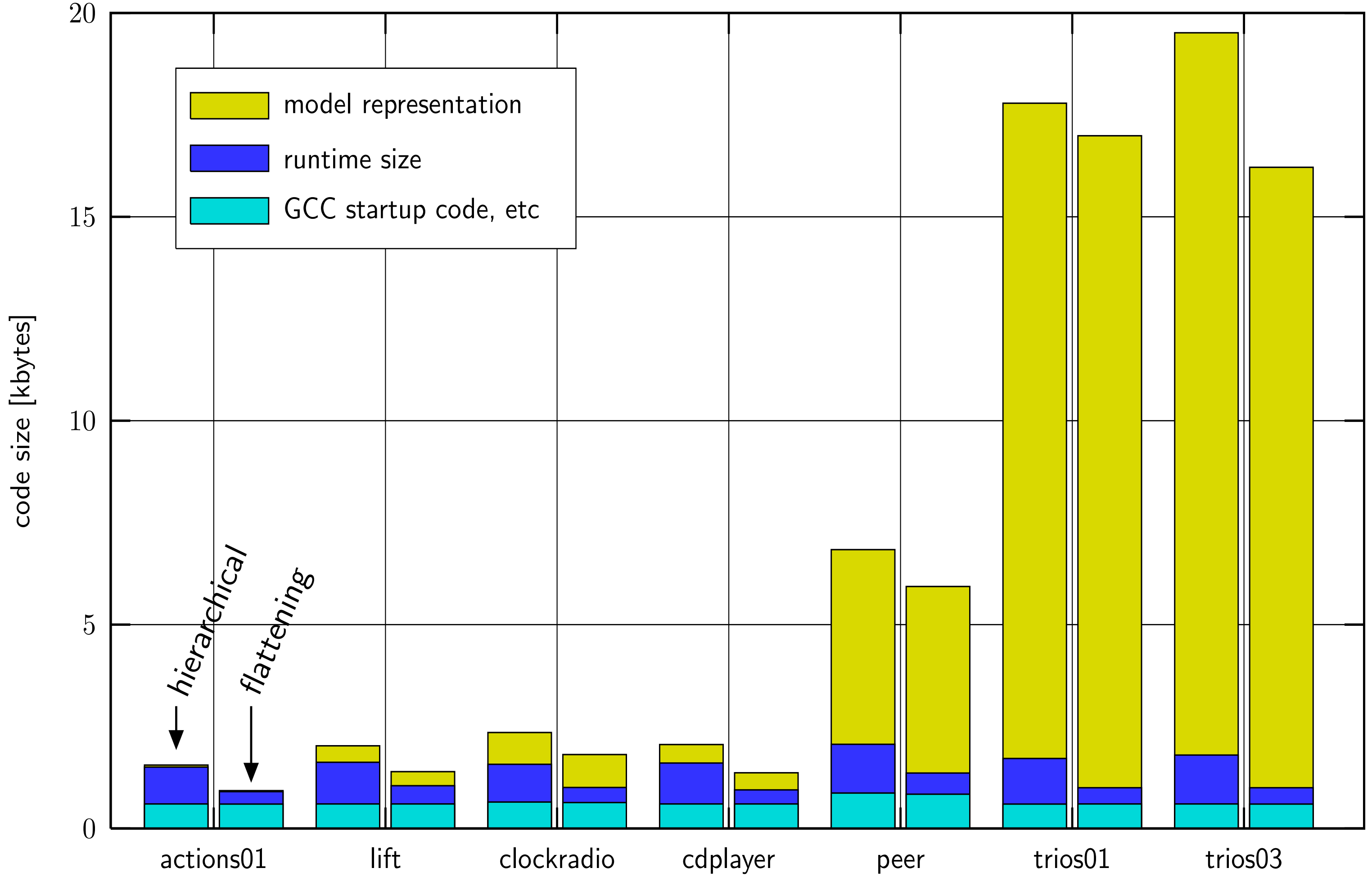




Outline

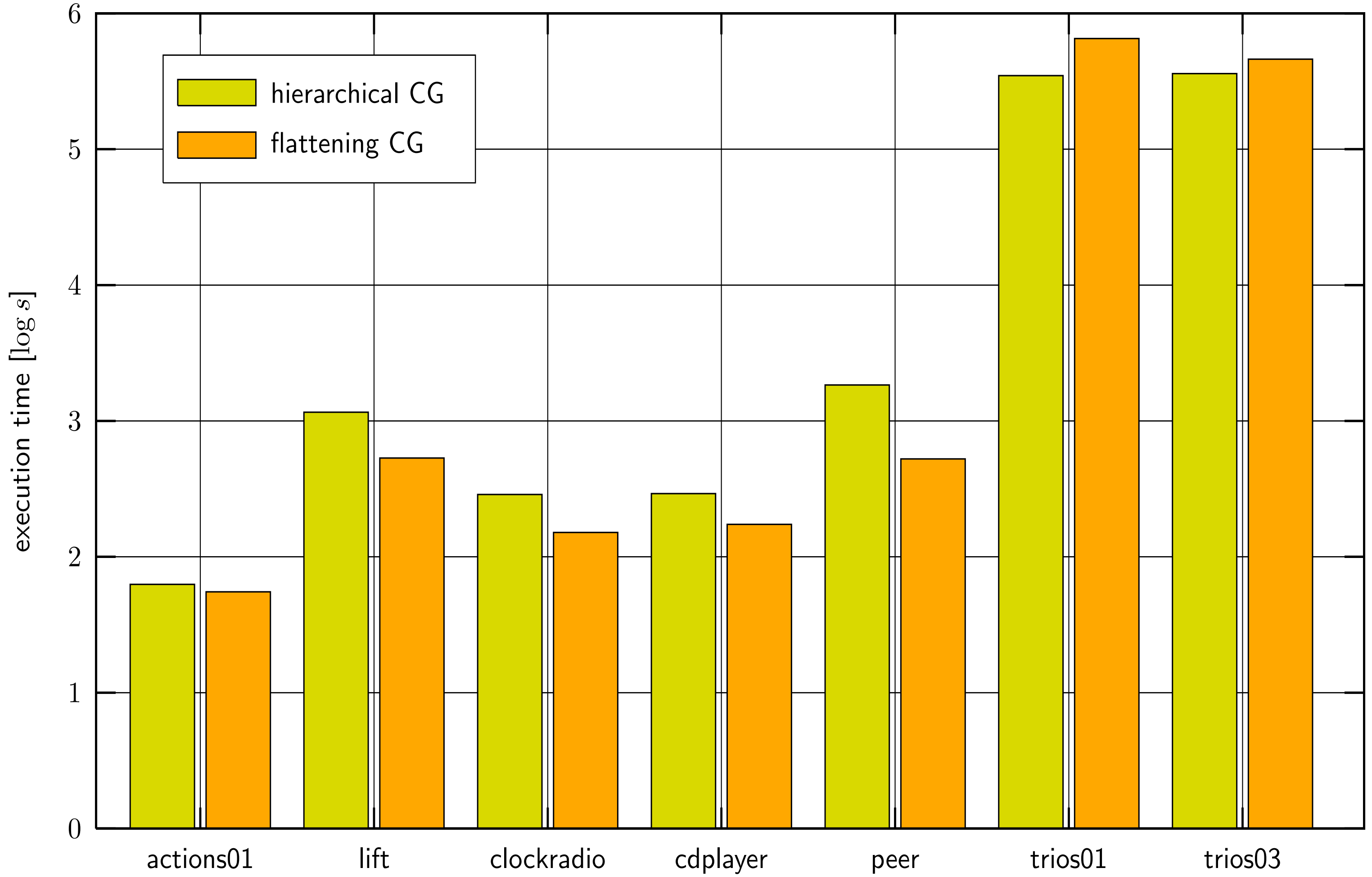
- **Introduction and Motivation**
  - Hierarchical Statecharts
  - Flat Statecharts
  - The Flattening problem
- **A Lower Bound for Flattening**
- **Polynomial flattening for Code Generation**
- **Experimental results**
- **Conclusion**

Experimental results



- Code size compiled for AVR using avr-gcc 3.3.2
- Bare model and runtime, dummy actions, dummy event feeder.
- Ratio varies from 0.6 to 0.96.

Experimental results (II)



- Execution time on pseudo-random trace of 10^7 events.
- Run on Linux 450MHz Pentium II.
- Ratio varies from 0.58 to 1.31.

Conclusion

- A superpolynomial, subexponential lower bound for flattening statecharts in absence of message passing, wchich indicates the expressive power of signal communication (sequencing).
- If your flattening does not rely on sequencing mechanism it is bound to explode.
- A polynomial algorithm for application in code generation, which is suprising (see at least 7 references for counter intuitions in the LCTES paper).
- An efficient code generator, which significantly improves over the industrial implementation.

Created with LaTeX, METAPOST, xfi g, gnuplot, awk & sed

Thank you for Your attention.