Distributed Synthesis

Jonathan Türpe

June 26, 2008

Introduction

What we did so far

[Andreas Augustin, Synthesis under incomplete information]

Synthesis problem

- Decide if a given specification has an implementation

Closed synthesis:

- single-process, no interaction with the environment

Open synthesis:

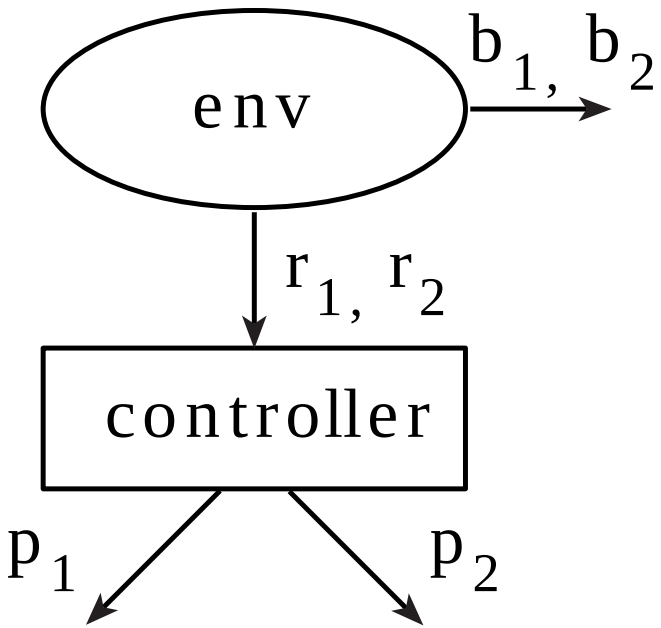
- single-process, interaction with the environment
- solved for CTL*

Open synthesis with incomplete information:

- single-process, interaction with the environment, hidden variables
- solved for CTL* [Kup.Var97]

Example: Incomplete information

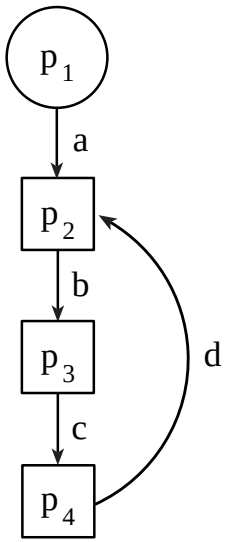
[Andreas Augustin, Synthesis under incomplete information]



Distributed synthesis

Distributed synthesis

- Multiple processes
- Interaction between the processes



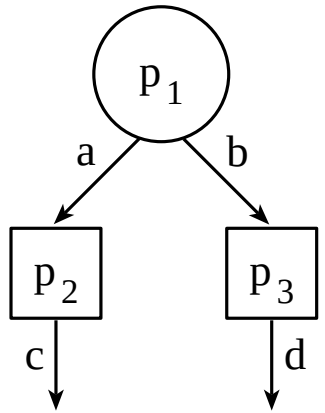
Distributed synthesis problem

- Given: A specification φ and an architecture A
- Decide if there are implementations such that the composition satisfies φ

History of distributed synthesis

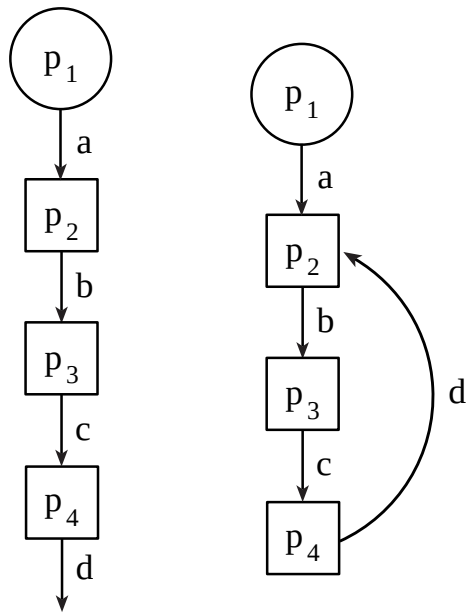
Negative result [Pnu.Ros90]

- Specification in LTL



Positive results [Kup.Var01]

- Specification in CTL*
- Pipeline-, ring architectures



Distributed synthesis

Generalized result [Fin.Sch05]

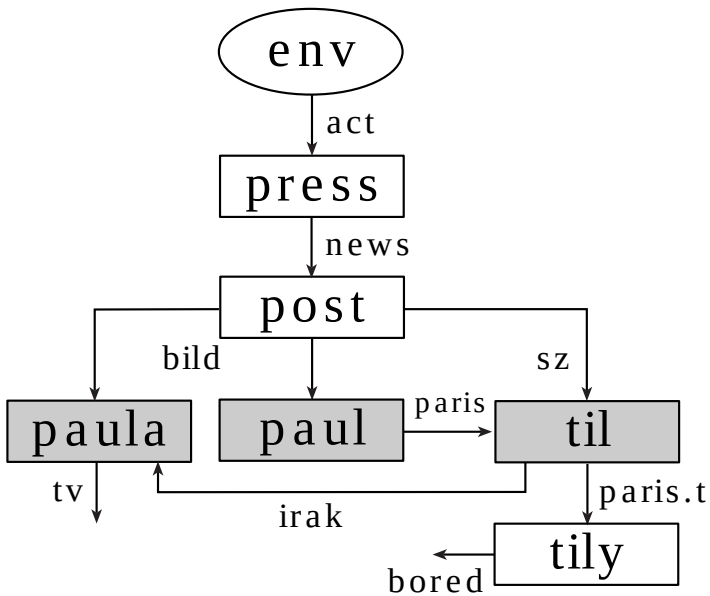
- Can partition architectures in two clases $\mathcal{D}$ and $\mathcal{U}$
- Positiv result
  - Specification φ in CTL*
  - Architectures in class $\mathcal{D}$
- Negativ result
  - Specification in LTL, or CTL
  - Architectures in class $\mathcal{U}$

Distributed synthesis

Definitions

Architecture A=(P, W, p_{env}, E, O, H)

- P set of processes
- $W \subset P$ white-box processes
- $p_{env} \in P \setminus W$
- (P, E) directed graph
- $O = \{O_e \mid e \in E\}$
- $H = \{H_p \mid p \in P\}$
- Notation: $B = P \setminus W$ for black-box processes



Definitions

Implementation of a process p

- Is a function $s_p : (2^{I_p})^* \longrightarrow 2^{O_p}$ called *strategy*
- I_p and O_p are the input or output variables of p

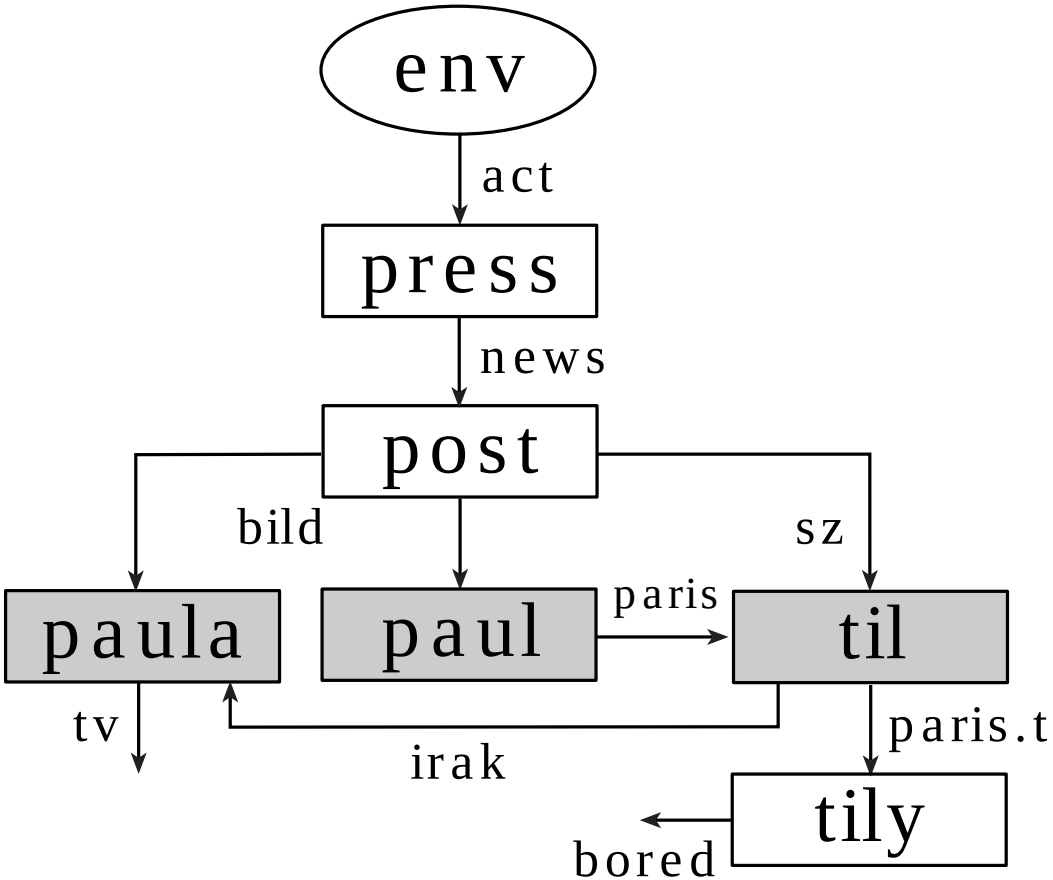
Implementation of an architecture A

- Set of strategies $S = \{s_p \mid p \in B \setminus p_{env}\}$

Composition of a set of strategies

- Function $s_Q : (2^{I_Q})^* \longrightarrow 2^{O_Q}$
- I_Q and O_Q is the common input or output respectively
- Mapping according to its single strategies

Example


env
act
press
news
post
bild
sz
paula
paul
paris
til
tv
irak
paris.t
tily
bored

Definitions

Computation tree of an implementation S

- Is a $2^{O \cup H}$ labeled $2^{O_{env}}$-tree
- $< 2^{O_{env}}{}^*, \ell >= xray_{2^{O_{env}}}(wide_{2^{H_{env}}}(< (2^{O_{env} \setminus H_{env}})^*, s_{B \setminus p_{env}} >))$

Definitions

Distributed synthesis problem

- Given a tuple (A, φ),
  consisting of an architecture A and a specification φ
- (A, φ) is *realizable* if there exists an implementation of A s.t. its computational tree satisfies φ.
- An architecture is called *decidable* if there exists an algorithm that decides for all specifications φ if (A, φ) is realizable.

Recapitulation of the definitions

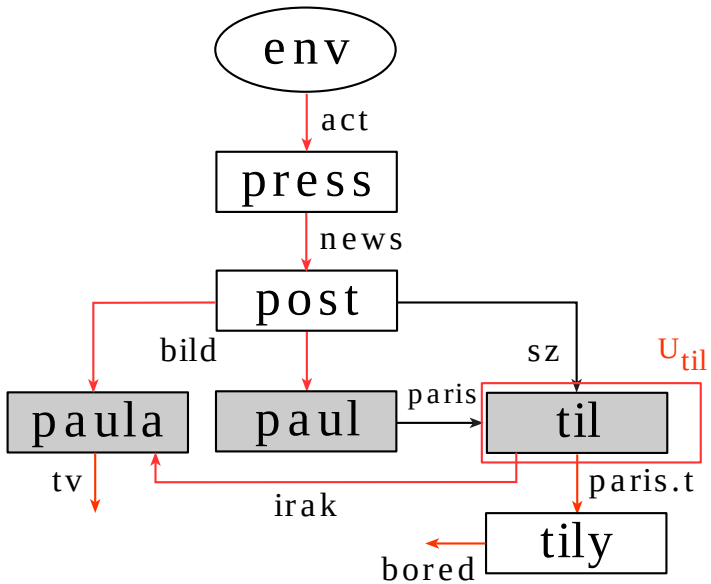
- Architecture
- Implementation
- Computation tree
- Distributed synthesis problem
- Decidability of an architecture

Partition the classes of architectures

Informedness of processes

Defining the preorder $p \preccurlyeq p'$

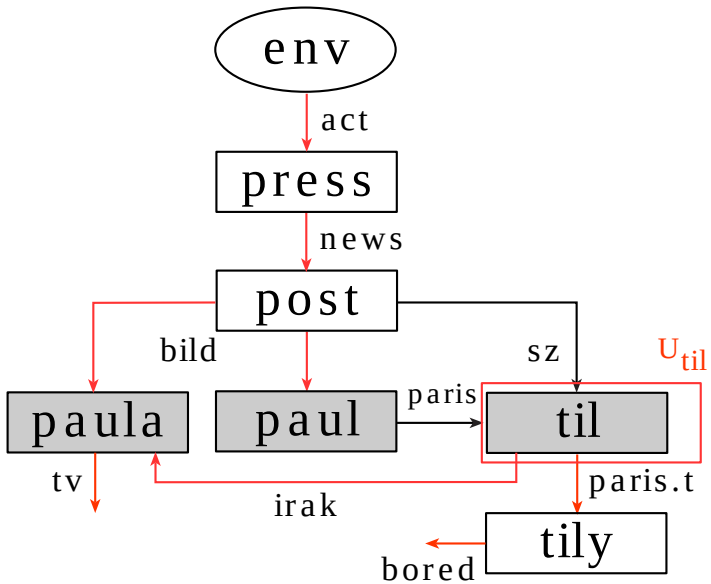
- Reading: p has more or equal information than p'



Informedness of processes

Defining the preorder $p \preccurlyeq p'$

- Reading: p has more or equal information than p'
- $E_p = \{e \in E \mid O_e \not\subseteq I_p\}$
- $U_p = \{q \in B \mid \neg\exists$ directed path from p_{env} to q in $(P, E_p)\}$
- $p \preccurlyeq p' \leftrightarrow p' \in U_p$ for $p, p' \in B$

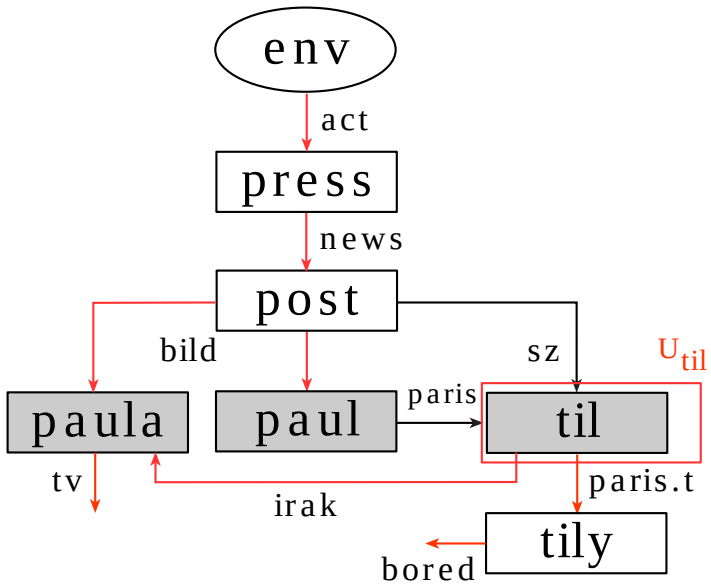


- E_p set of edges with information invisible to p
- U_p processes that are not reachable by this edges

Informedness of processes

Defining the preorder $p \preccurlyeq p'$

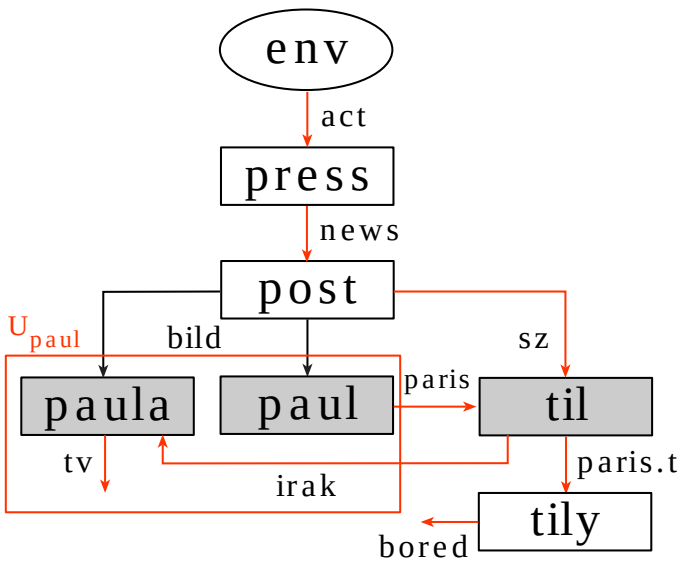
- Reading: p has more or equal information than p'
- $E_p = \{e \in E \mid O_e \not\subseteq I_p\}$
- $U_p = \{q \in B \mid \neg\exists$ directed path from p_{env} to q in $(P, E_p)\}$
- $p \preccurlyeq p' \leftrightarrow p' \in U_p$ for $p, p' \in B$



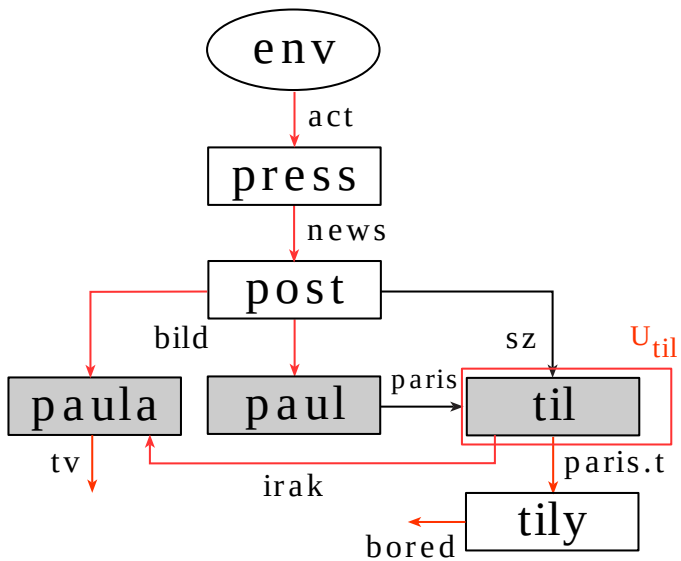
- E_p set of edges with information invisible to p
- U_p processes that are not reachable by this edges
- *Example* : $til \preccurlyeq til$

Example

$paul \not\leq til$



$til \not\leq paul$



Informdness of processes

An architecture A is called *ordered*

- Exists a surjective function $f : B \to \mathbb{N}_n$
- $f^{-1}(1) = \{p_{env}\}$
- For all $p, p' \in B : f(p) \leq f(p')$ iff. $p \preccurlyeq p'$
  i.e. p has more or equal information than p'

Informdness of processes

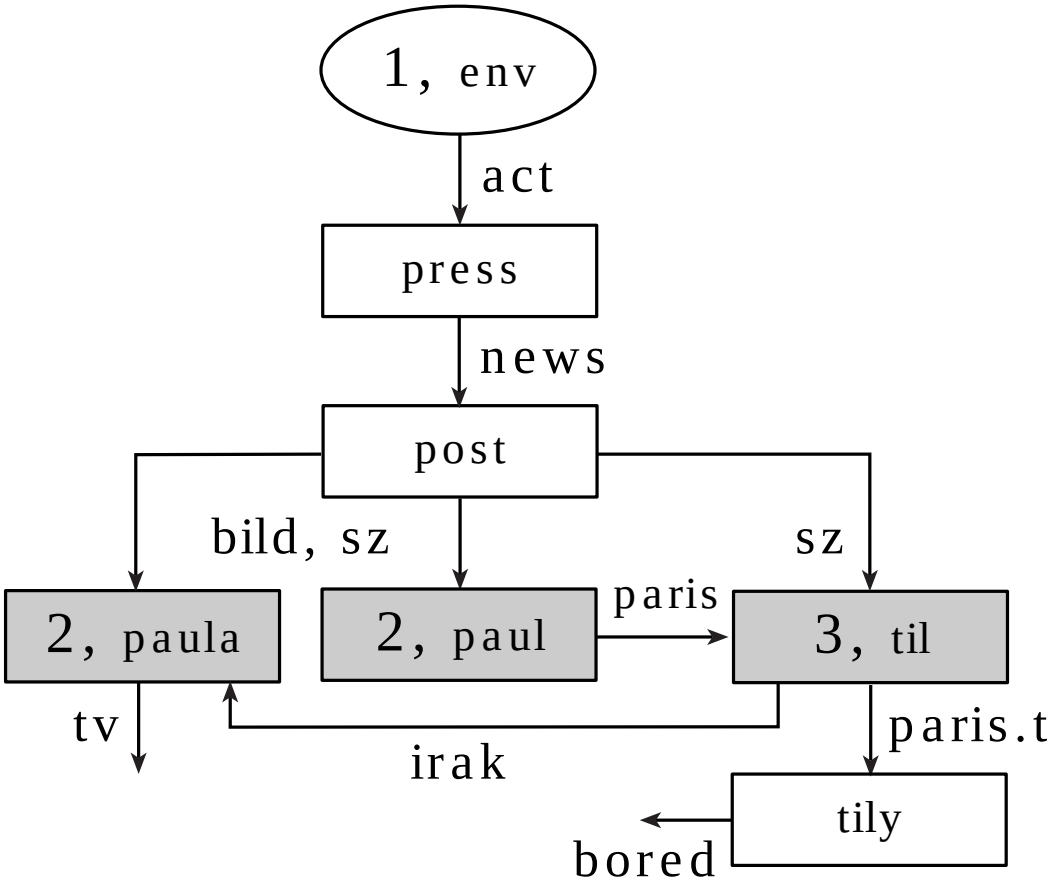
An architecture A is called *strictly ordered*

- Exists a bijective function $f : B \to \mathbb{N}_n$
- $f^{-1}(1) = \{p_{env}\}$
- For all $p, p' \in B : f(p) \leq f(p')$ iff $p \preccurlyeq p'$
  i.e. p has more or equal information than p'

$\mathcal{D}$ and $\mathcal{U}$

- An architecture A is in $\mathcal{D}$ if A can be ordered
- An architecture A is in $\mathcal{U}$ otherwise

Example


1, env
act
press
news
post
bild, sz
sz
2, paula
2, paul
paris
3, til
tv
irak
paris.t
tily
bored

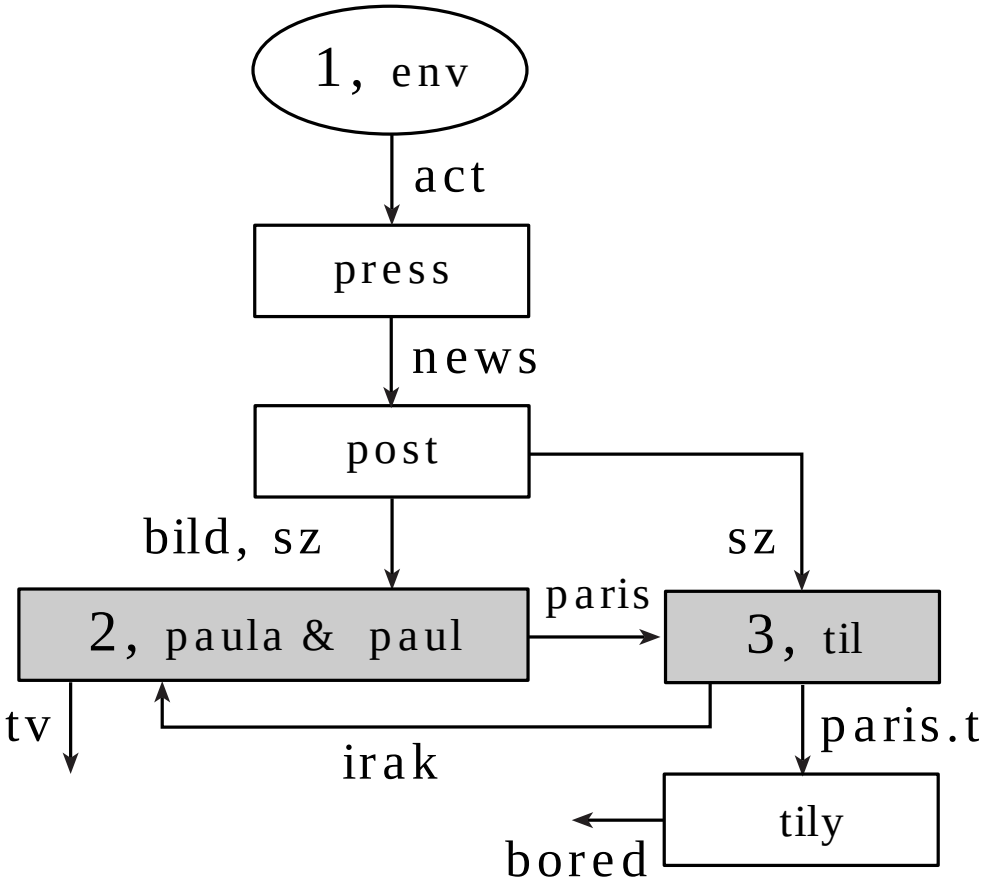
Decidability of ordered architectures

Architecture transformation

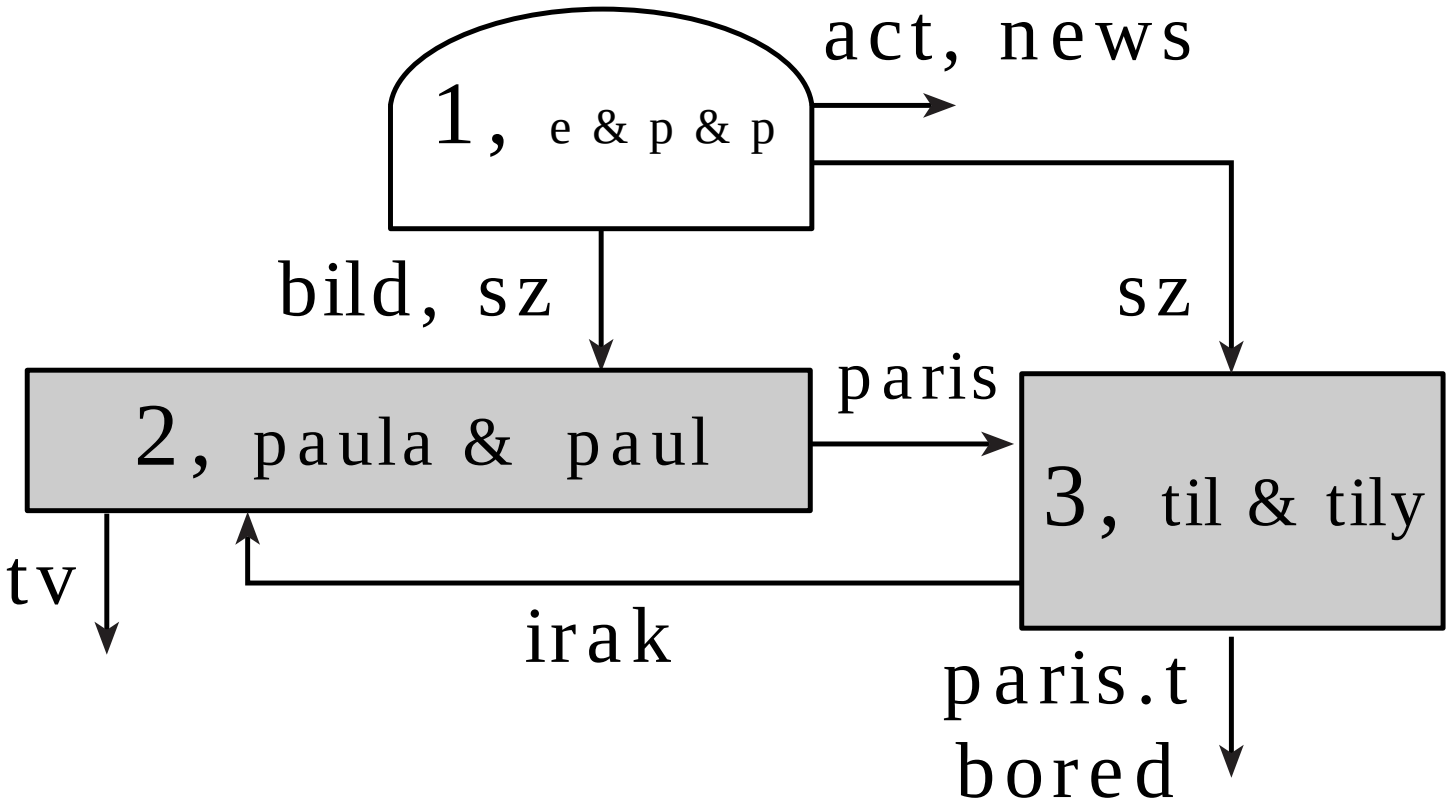
Consider only ordered architectures A.
We can equivalently transform the decidability problem from (A, φ) to (A', φ').

1. Clustering equally informed processes
2. Elimination of white-box processes
3. Elimination of feedback edges

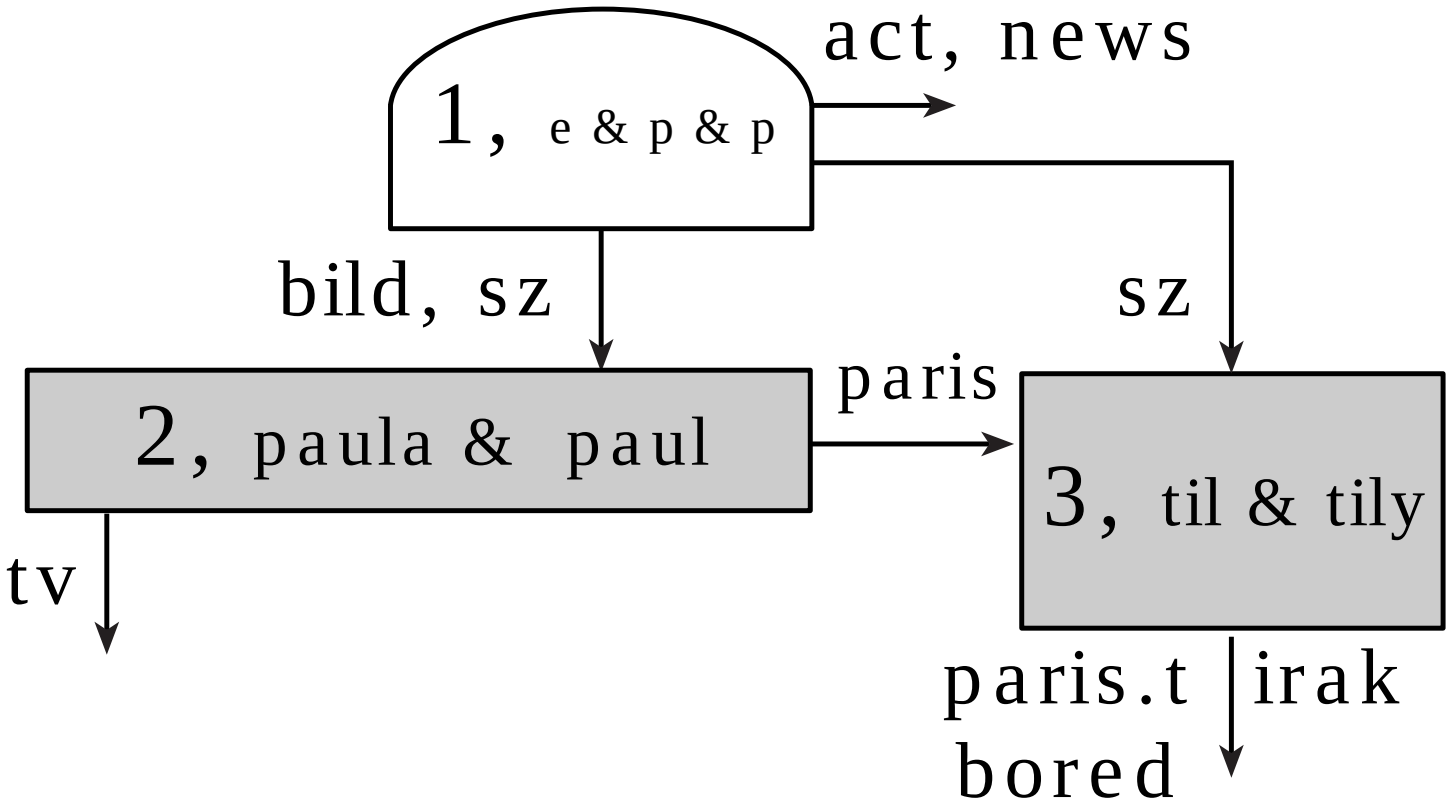
Clustering equally informed processes



Elimination of white-box processes



Elimination of feedback edges



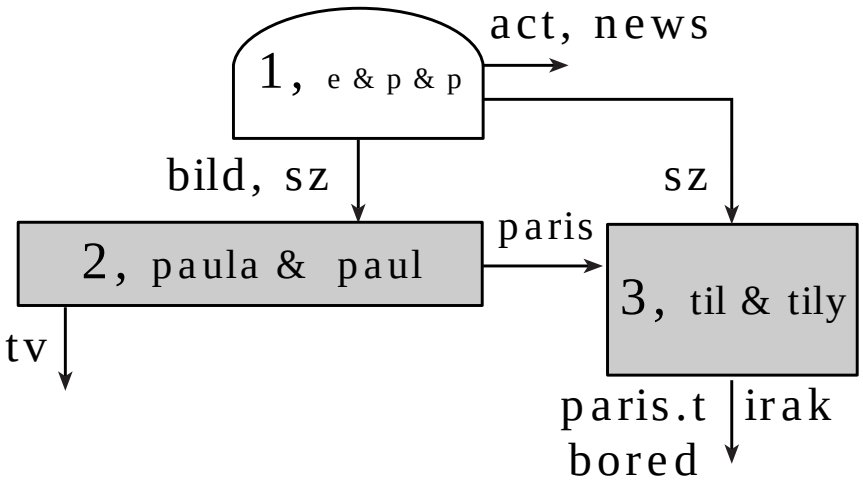
Automaton construction

Idea:

Given the transformed distributed synthesis problem (A', φ').
(A' is an strictly ordered architecture with no white-boxes and no feedback edges and φ' a CTL*-formula)

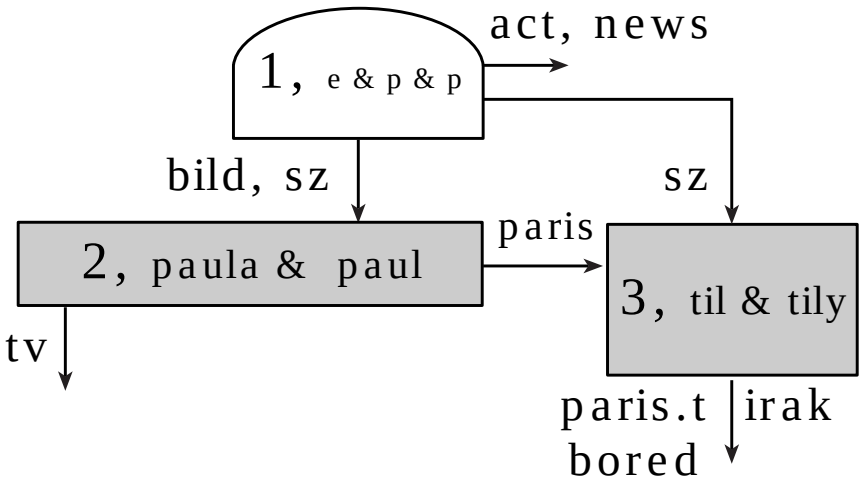
Construct an alternating automaton $\mathcal{A}$ that recognizes a tree iff it is a computational tree of A' that satisfies φ'.

Automaton construction



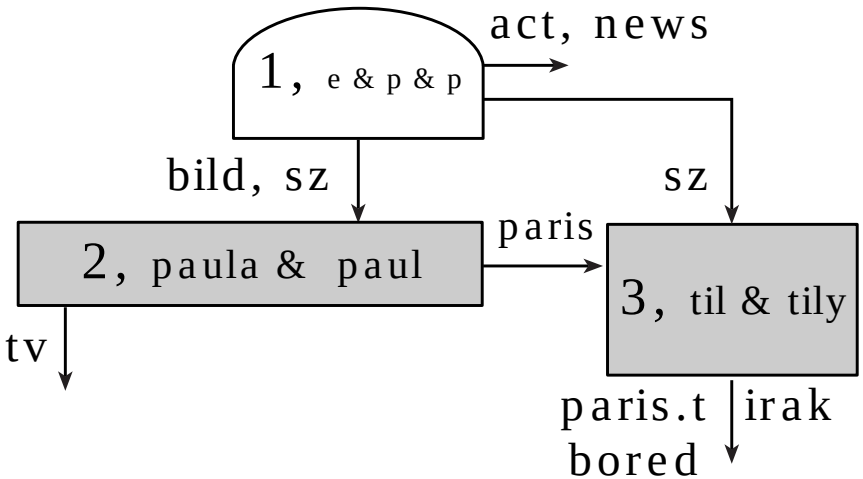
1. $\varphi \rightsquigarrow \mathcal{A}_1$ over $2^{\{a,n,b,s,t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees

Automaton construction



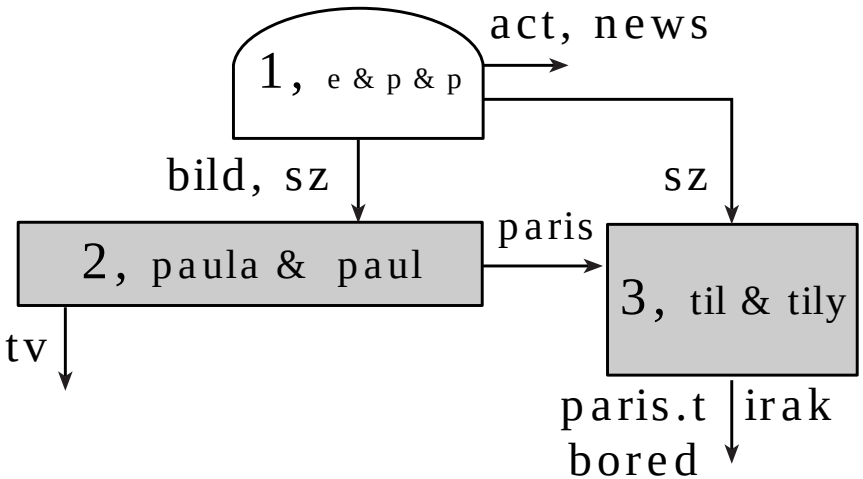
1. $\varphi \rightsquigarrow \mathcal{A}_1$ over $2^{\{a,n,b,s,t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
2. $\mathcal{A}_1 \rightsquigarrow \mathcal{A}_2$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees

Automaton construction



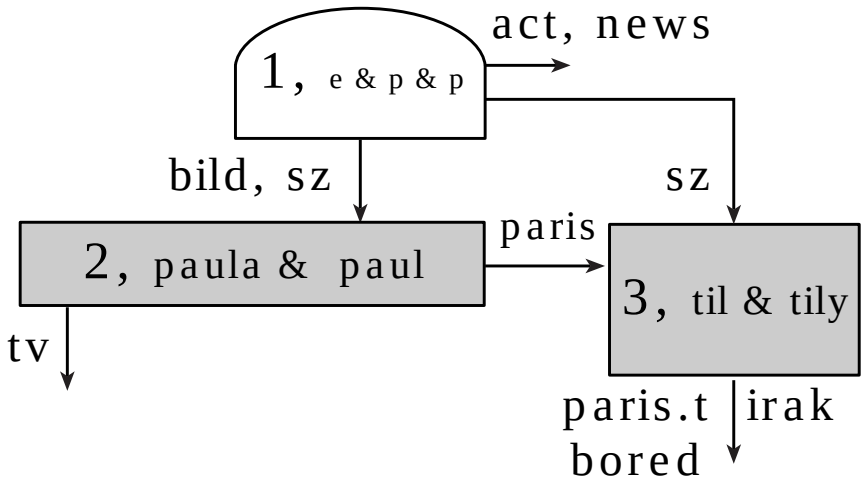
1. $\varphi \rightsquigarrow \mathcal{A}_1$ over $2^{\{a,n,b,s,t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
2. $\mathcal{A}_1 \rightsquigarrow \mathcal{A}_2$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
3. $\mathcal{A}_2 \rightsquigarrow \mathcal{A}_3$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{b,s\}}$-trees

Automaton construction



1. $\varphi \rightsquigarrow \mathcal{A}_1$ over $2^{\{a,n,b,s,t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
2. $\mathcal{A}_1 \rightsquigarrow \mathcal{A}_2$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
3. $\mathcal{A}_2 \rightsquigarrow \mathcal{A}_3$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{b,s\}}$-trees
4. $\mathcal{A}_3 \rightsquigarrow \mathcal{N}_3$ nondeterministic automaton

Automaton construction



1. $\varphi \rightsquigarrow \mathcal{A}_1$ over $2^{\{a,n,b,s,t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
2. $\mathcal{A}_1 \rightsquigarrow \mathcal{A}_2$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{a,n,b,s\}}$-trees
3. $\mathcal{A}_2 \rightsquigarrow \mathcal{A}_3$ over $2^{\{t,p,p',i,b\}}$-labeled $2^{\{b,s\}}$-trees
4. $\mathcal{A}_3 \rightsquigarrow \mathcal{N}_3$ nondeterministic automaton
5. $\mathcal{N}_3 \rightsquigarrow \mathcal{A}_4$ over $2^{\{p',i,b\}}$-labeled $2^{\{b,s\}}$-trees

Decidability of $\mathcal{D}$

Theorem.
Let A be an ordered architecture and φ a CTL*-formula.

The distributed synthesis problem for (A, φ) is decidable.

Undecidability of architectures that can not be ordered

Undecidability of $\mathcal{A}_2$

Theorem. Let A be an architecture, s.t.
A can not be ordered and φ a CTL or LTL specification.

The distributed synthesis problem for (A, φ) is undecidable.

Proof idea: Using a reduction from the halting problem.

Thank you for your attention.

[Fin.Sch05] Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: Proc. LICS, IEEE Computer Society Press (2005) 321- 330

[Kup.Var97] O. Kupferman and M. Y. Vardi. Synthesis with incomplete information. In Proc. ICTL'97, 1997.

[Kup.Var01] O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In Proc. LICS'01, July 2001.

[Pnu.Ros90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In Proc. FOCS'90, pages 746-757, 1990.