A Basic Parallel Process as a Parallel Pushdown Automaton

Paul van Tilburg

(joint work with Jos Baeten and Pieter Cuijpers)

Department of Mathematics and Computer Science
Eindhoven University of Technology

EXPRESS ’08
Toronto, Canada / August 23, 2008

Project MoCAP

- Models of Computation: Automata and Processes

Automata + *Interaction* = Concurrency

Technische Universiteit Eindhoven University of Technology

Introduction

Project MoCAP

- Models of Computation: Automata and Processes

Automata + *Interaction* = Concurrency

- Context-free process as a pushdown automaton [CONCUR'08]
- Study similarities and differences
- Different approach

Regular Language

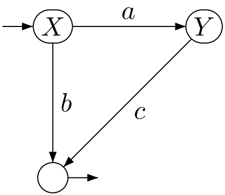
Right-linear grammar

Generates a regular language

$$X \longrightarrow aY \mid b$$
$$Y \longrightarrow c$$

Non-deterministic finite automaton

Accepts a regular language



Also: (finite) transition system

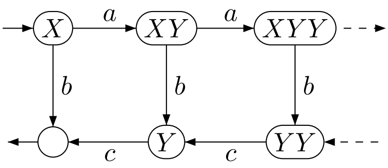
Context-Free Language

Context-free grammar

Generates a context-free language

$$X \longrightarrow aXY \mid b$$
$$Y \longrightarrow c$$

Transition system



Famous theorem from automata theory

For every context-free language there exists a pushdown automaton that accepts it.

Technische Universiteit Eindhoven University of Technology

Processes and Recursive Specifications

Context-free grammar

Generates a context-free language

$$X \longrightarrow aXY \mid b$$
$$Y \longrightarrow c$$

Recursive specification over $\mathrm{BPA}_{\mathbf{0},\mathbf{1}}$

Specifies a context-free process

$$X = a.(X \cdot Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Restrict to:
finite and guarded specifications

Technische Universiteit Eindhoven University of Technology

Processes and Recursive Specifications

Context-free grammar

Generates a context-free language

$$X \longrightarrow aXY \mid b$$
$$Y \longrightarrow c$$

Recursive specification over $\mathrm{BPA}_{\mathbf{0},\mathbf{1}}$

Specifies a context-free process

$$X = a.(X \cdot Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Restrict to:
finite and guarded specifications

$\mathbf{0}$ and $\mathbf{1}$

- Used to express deadlocked state ($\mathbf{0}$) and final state ($\mathbf{1}$)

Technische Universiteit Eindhoven University of Technology

Theorem for Context-Free Processes

Process theory enables us to introduce *interaction* by. . .

- Modeling the data (a stack) as a process
- Making communication with the stack explicit
- Using bisimulation equivalences to preserve branching structure

Theorem

Every context-free process is equivalent to a regular process communicating with a stack. [CONCUR'08]

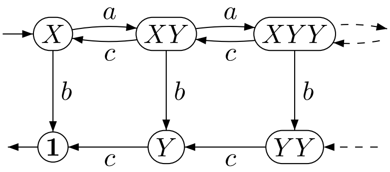
Basic Parallel Processes

Recursive specification over $\mathrm{BPP}_{\mathbf{0},\mathbf{1}}$

Specifies a basic parallel process

$$X = a.(X \parallel Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Transition system



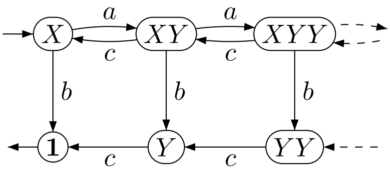
Basic Parallel Processes

Recursive specification over $\text{BPP}_{\mathbf{0,1}}$

Specifies a basic parallel process

$$X = a.(X \parallel Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Transition system



Theorem

Every basic parallel process is equivalent to a regular process communicating with a bag.

Technische Universiteit Eindhoven University of Technology

Specification over $\mathrm{BPA}_{\mathbf{0},\mathbf{1}}$

$$B = \mathbf{1} + \sum_{d \in D} ?_i d.(B \parallel !_o d.\mathbf{1})$$

Interaction

Use $\gamma(!_c d, ?_c d) = !?_c d$ for all $d \in D$ and channel $c = i, o$

$$!_i d.P \parallel_\gamma B \xrightarrow{!?_i d} P \parallel_\gamma (B \parallel !_o d.\mathbf{1})$$

$$?_o d.P \parallel_\gamma (B \parallel !_o d.\mathbf{1}) \xrightarrow{!?_o d} P \parallel_\gamma B$$

TU/e Technische Universiteit Eindhoven University of Technology

Parallel Pushdown Automaton

Basic parallel process

$$X = a.(X \parallel Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Translated

$$\hat{X} = a.\mathrm{Push}(XY) + b.\mathrm{Push}(\emptyset)$$
$$\hat{Y} = c.\mathrm{Push}(\emptyset)$$

$$\mathrm{Push}(\emptyset) = \mathrm{Ctrl}$$
$$\mathrm{Push}(X\xi) = !_i X.\mathrm{Push}(\xi)$$
$$\mathrm{Ctrl} = \sum_{V \in \mathcal{V}} ?_o V.\hat{V}$$

in parallel with a bag:

$$B = \mathbf{1} + \sum_{V \in \mathcal{V}} ?_i V.(B \parallel !_o V.\mathbf{1})$$

Technische Universiteit Eindhoven University of Technology

Parallel Pushdown Automaton

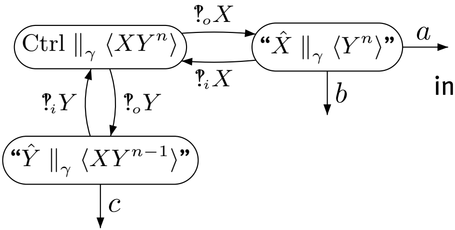
Basic parallel process

$$X = a.(X \parallel Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1}$$

Translated

$$\hat{X} = a.\mathrm{Push}(XY) + b.\mathrm{Push}(\emptyset)$$
$$\hat{Y} = c.\mathrm{Push}(\emptyset)$$

$$\mathrm{Push}(\emptyset) = \mathrm{Ctrl}$$
$$\mathrm{Push}(X\xi) = !_i X.\mathrm{Push}(\xi)$$
$$\mathrm{Ctrl} = \sum_{V \in \mathcal{V}} ?_o V.(\hat{V} + !_i V.\mathrm{Ctrl})$$



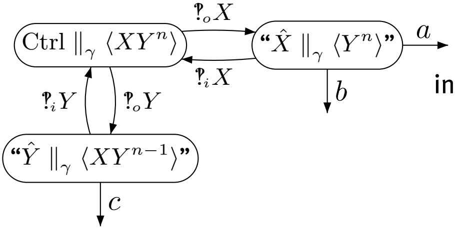
in parallel with a bag:

$$B = \mathbf{1} + \sum_{V \in \mathcal{V}} ?_i V.(B \parallel !_o V.\mathbf{1})$$

Parallel Pushdown Automaton

Basic parallel process

$$X = a.(X \parallel Y) + b.\mathbf{1}$$
$$Y = c.\mathbf{1} + \mathbf{1}$$



Translated

$$\hat{X} = a.\text{Push}(XY) + b.\text{Push}(\emptyset)$$
$$\hat{Y} = c.\text{Push}(\emptyset) + \mathbf{1}$$

$$\text{Push}(\emptyset) = \text{Ctrl}$$
$$\text{Push}(X\xi) = !_i X.\text{Push}(\xi)$$
$$\text{Ctrl} = \sum_{V \in \mathcal{V}} ?_o V.(\hat{V} + !_i V.\text{Ctrl})$$

in parallel with a partially forgetful bag:

$$B = \mathbf{1} + \sum_{V \in \mathcal{V} - \mathcal{V}^{+1}} ?_i V.(B \parallel !_o V.\mathbf{1}) + \sum_{V \in \mathcal{V}^{+1}} ?_i V.(B \parallel (!_o V.\mathbf{1} + \mathbf{1}))$$

Technische Universiteit Eindhoven University of Technology

Results & Concluding Remarks

Proved Theorem

For every basic parallel process P *there exists a regular process* Q *such that* $P = \tau_*(\partial_*(Q \parallel_\gamma B))$.

- Solution modulo rooted branching bisimulation
- Made communication with the bag explicit
- The (partially forgetful) bag is the prototypical basic parallel process

TU/e Technische Universiteit Eindhoven University of Technology

Results & Concluding Remarks

Proved Theorem

For every basic parallel process P *there exists a regular process* Q *such that* $P = \tau_*(\partial_*(Q \parallel_\gamma B))$.

- Solution modulo rooted branching bisimulation
- Made communication with the bag explicit
- The (partially forgetful) bag is the prototypical basic parallel process

Corollary

Every basic parallel process has bounded branching.

Technische Universiteit Eindhoven University of Technology

Results & Concluding Remarks

Proved Theorem

For every basic parallel process P *there exists a regular process* Q *such that* $P = \tau_*(\partial_*(Q \parallel_\gamma B))$.

- Solution modulo rooted branching bisimulation
- Made communication with the bag explicit
- The (partially forgetful) bag is the prototypical basic parallel process

Corollary

Every basic parallel process has bounded branching.

Future work

- Reverse case, maybe with 1?
- Queues?

TU/e Technische Universiteit Eindhoven University of Technology

Thank you!

Questions?