CWI

Parallel Pushdown Automata and Commutative Context-Free Grammars in Bisimulation Semantics

Jos Baeten

fellow CWI

emeritus UvA & TU/e

OPCT, Bertinoro, 27 June 2023

CWI

Draft paper

- Parallel Pushdown Automata and Commutative Context-Free Grammars in Bisimulation Semantics
- with Bas Luttik (Eindhoven University of Technology)
- to be submitted to EXPRESS/SOS 2023, Antwerp

CWI

What is a computation?

- Church-Turing Thesis
- Given by a Turing machine: input at begin, deterministic steps, output at end
- A computation is a function
- Models a deaf, dumb and blind computer (before advent of terminal)
- Far removed from a modern-day computer as students see all around them

CWI

Reactive Systems

“A Turing machine cannot drive a car, but a real computer can!”

CWI

Interaction

User interaction: not just initial, final word on the tape.

Make interaction between control and memory explicit.

Integration of automata theory and process theory.

Aim to develop course on foundations of computer science for all first-year computer science students.
Syllabus: Models of Computation based on Automata: Formal Languages and Communicating Processes

CWI

Reactive Turing Machine

Defined in I&C 2013 with Bas Luttik and Paul van Tilburg

Executability instead of computability, but not more expressive

Robustness: π-calculus (Robin Milner) is the λ-calculus with interaction (with Bas Luttik and Fei Yang)

CWI

Pushdown Automata and Context-Free Grammars in Bisimulation Semantics

- with Cesare Carissimo and Bas Luttik
- arXiv:2203.01713
- LMCS 2023

CWI

Well-known theorem

A language is defined by a pushdown automaton iff it is defined by a context-free grammar.

CWI

Well-known theorem

A language is defined by a pushdown automaton iff it is defined by a context-free grammar.

A process is defined by a pushdown automaton iff it is defined by a finite guarded recursive specification over a process algebra with actions, choice and sequencing including sequential value passing.

CWI

Well-known theorem

A language is defined by a pushdown automaton iff it is defined by a context-free grammar.

A process is defined by a pushdown automaton iff it is defined by a finite guarded recursive specification over a process algebra with actions, choice and sequencing including sequential value passing.

What if we replace the stack of the pda by a bag, and value passing sequencing by parallel composition with value passing communication?

CWI

Process

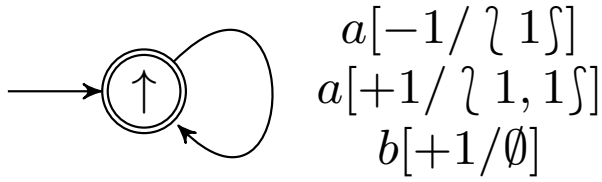
Labelled transition system $(\mathcal{S}, \mathcal{A}, \rightarrow, \downarrow)$

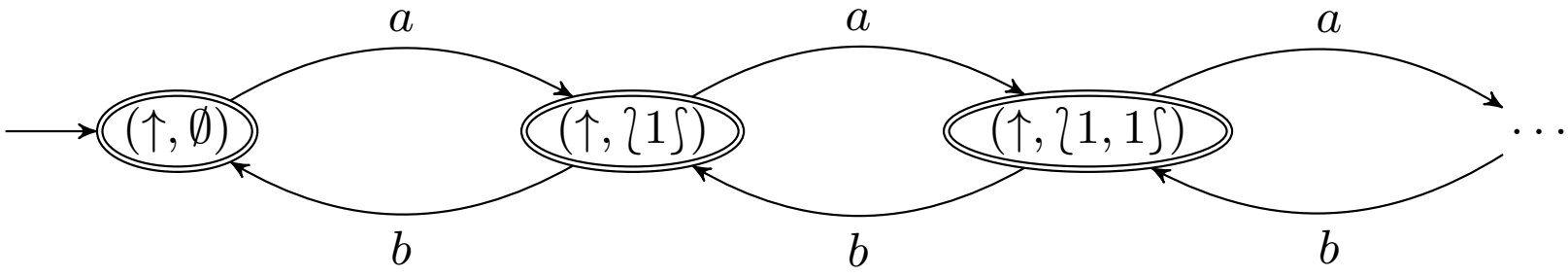
1. $\mathcal{S}$ is a set of *states*;
2. $\mathcal{A}$ is a set of *actions*, $\tau \notin \mathcal{A}$ is the *unobservable* or *silent* action;
3. $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \cup \{\tau\} \times \mathcal{S}$ is a *transition relation*
4. $\downarrow \subseteq \mathcal{S}$ is the set of *accepting* states

A *process graph* is a labelled transition system with a root state $\uparrow$.
Strong bisimulation $\leftrightarroweq$, branching bisimulation $\leftrightarroweq_{\mathsf{b}}$,
divergence-preserving branching bisimulation $\leftrightarroweq_{\mathsf{b}}^{\Delta}$.
A *process* is a $\leftrightarroweq_{\mathsf{b}}^{\Delta}$-equivalence class of process graphs.

CWI

Parallel Pushdown Automaton

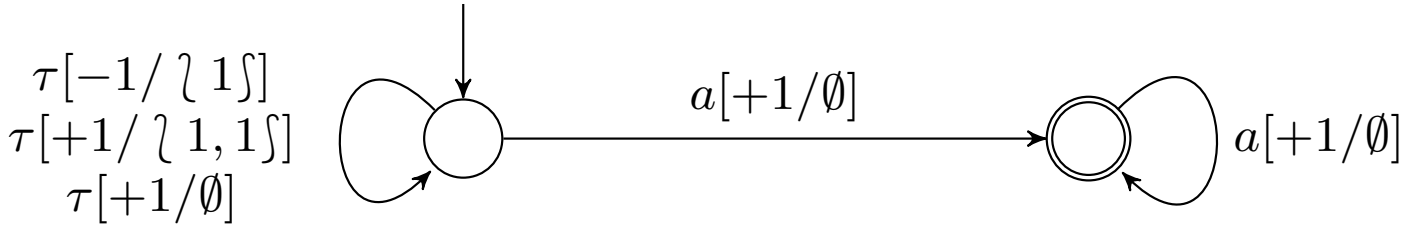


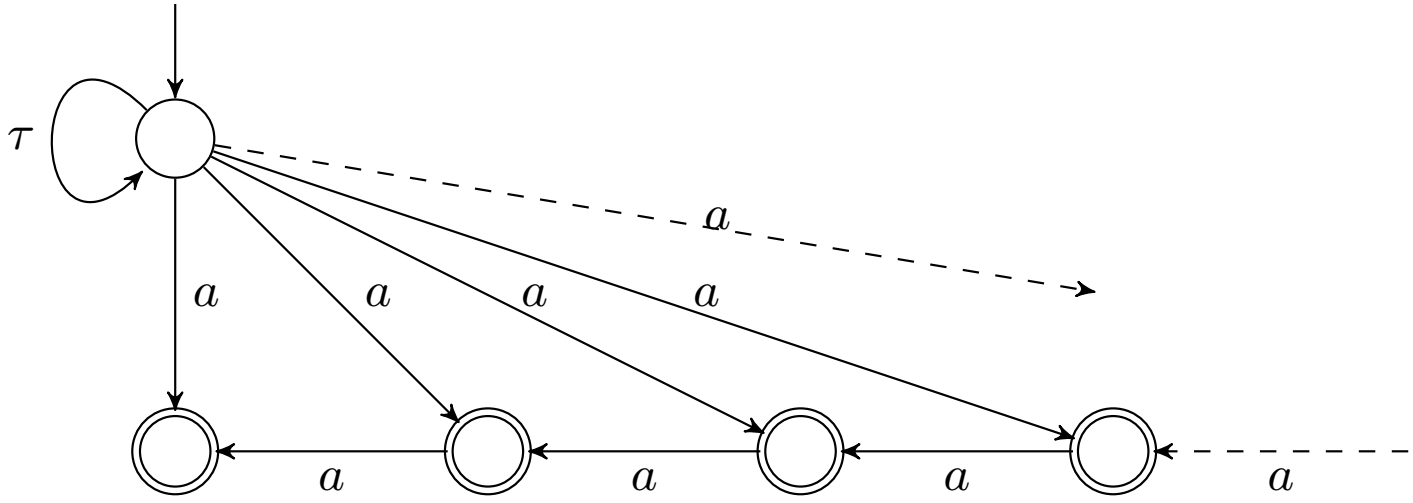


Insert and remove. Bounded branching under $\leftrightarrows$. Acceptance by final state can express more than acceptance by empty bag.

CWI

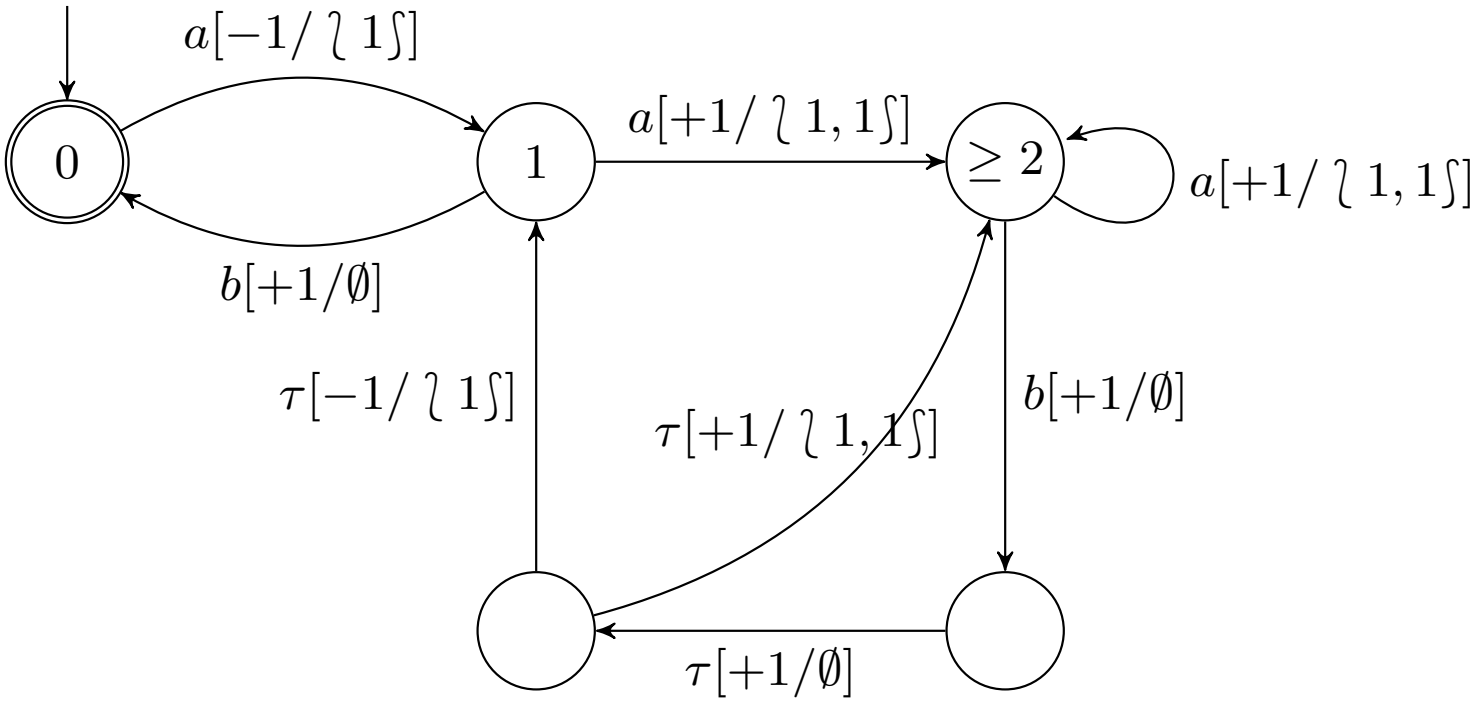
Bounded branching?





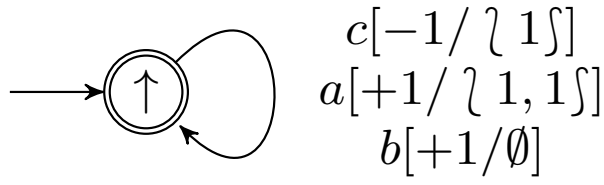
CWI

Acceptance by empty bag.



CWI

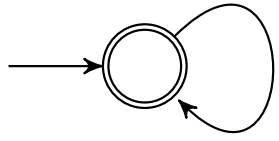
Counter slightly changed.



The c can be executed only when no b can be executed any longer.
Executing a b takes priority to executing a c.

CWI

Bag always accepting.



$ins(d)[-d/\wr\, d\int]$
$ins(d)[+d/\wr\, d, d\int]$
$rem(d)[+d/\emptyset]$
$show(^{-}d)[-d/\emptyset]$

Again, a $[-d/x]$ transition can only occur when a $[+d/y]$ transition can no longer occur. The remove transition should take priority over the show absence transition.

CWI

Basic Parallel Processes

Use SOS to give automata for syntax $\mathbf{0}, \mathbf{1}, a., +, \|$

$$\frac{}{\mathbf{1}\downarrow} \qquad \frac{}{a.p \xrightarrow{a} p}$$

$$\frac{p\downarrow}{(p+q)\downarrow} \qquad \frac{q\downarrow}{(p+q)\downarrow} \qquad \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$$

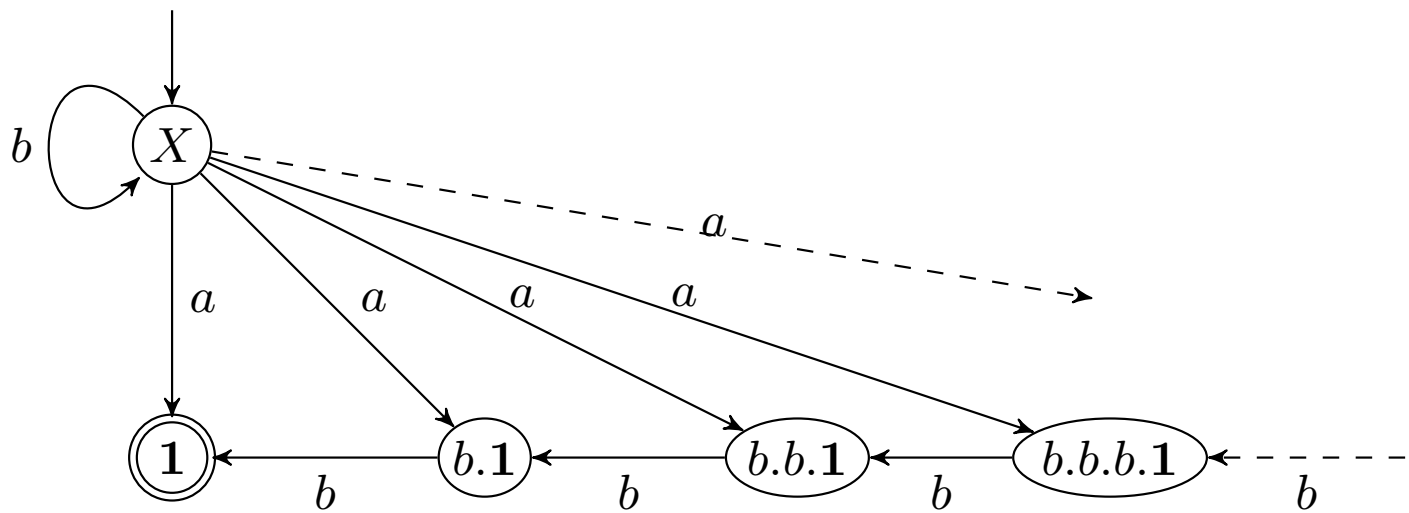
$$\frac{p\downarrow \quad q\downarrow}{p \parallel q\downarrow} \qquad \frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad \frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'}$$

$$\frac{p \xrightarrow{a} p' \quad X \stackrel{\text{def}}{=} p}{X \xrightarrow{a} p'} \qquad \frac{p\downarrow \quad X \stackrel{\text{def}}{=} p}{X\downarrow}$$

CWI

Guardedness necessary

$$X \stackrel{\text{def}}{=} a.\mathbf{1} + X \parallel b.\mathbf{1} \ .$$



Not divergence-preserving branching bisimilar to the process graph of a parallel pushdown automaton

CWI

BPP to CCFG

Theorem. Every weakly guarded recursive specification has a process graph that is divergence-preserving branching bisimilar to the process graph of a parallel pushdown automaton.

Proof. Every process expression p can be brought into basic parallel normal form $p \leftrightarrows (\mathbf{1}+) \sum_{i=1}^{n} a_i.p_i$, using

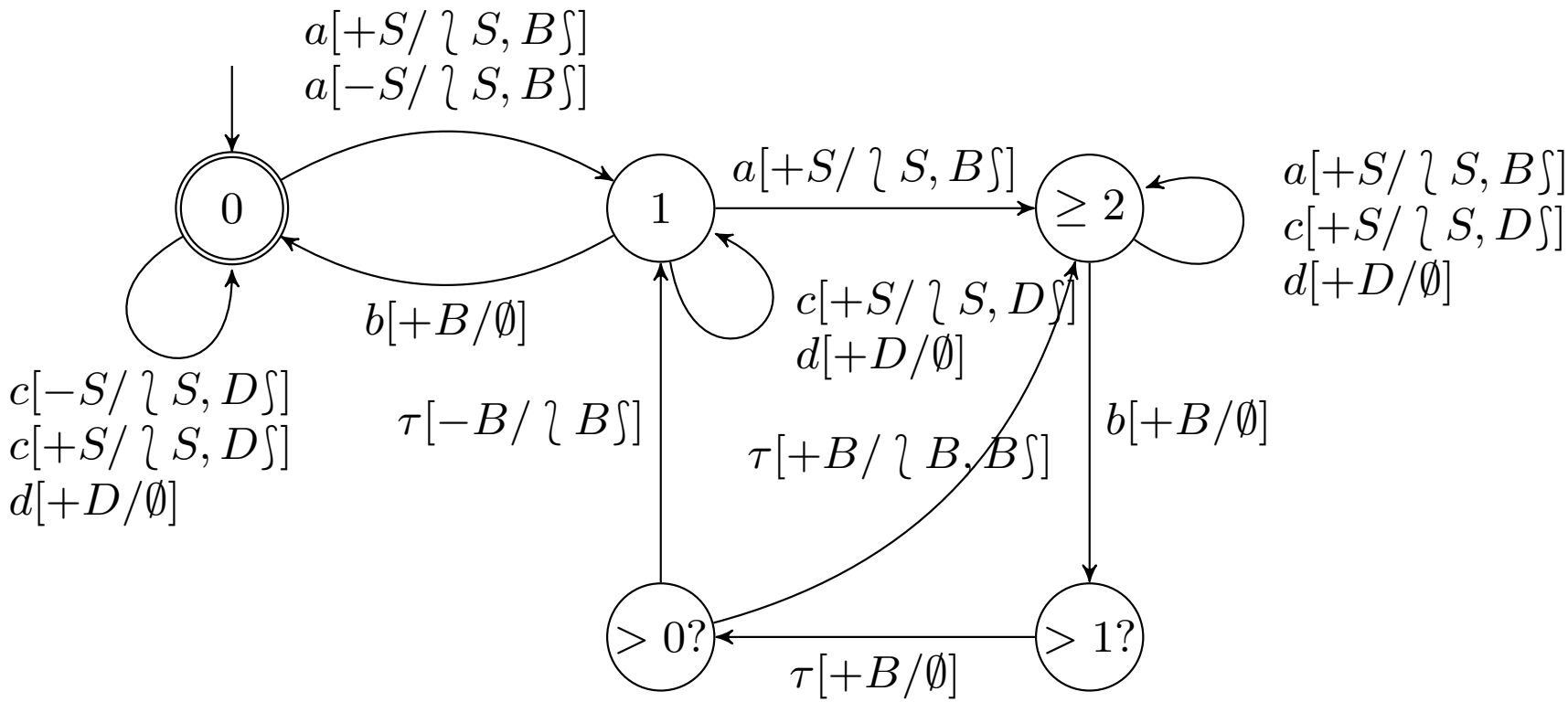
$$(\sum_{i=1}^{n} a_i.p_i) \parallel (\sum_{j=1}^{m} b_j.q_j) \leftrightarrows \sum_{i=1}^{n} a_i.(p_i \parallel q) + \sum_{j=1}^{m} b_j.(p \parallel q_j)$$

A weakly guarded specification can be brought with $\leftrightarrows$ into Greibach Normal Form $X \stackrel{\text{def}}{=} (\mathbf{1}+) \sum_{i=1}^{n} a_i.\xi_i$ (ξ_i a parallel composition of identifiers).

CWI

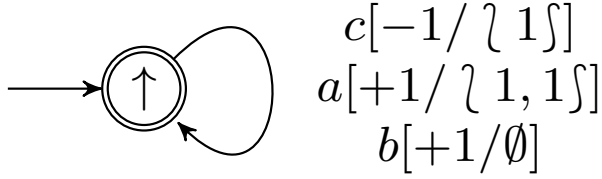
BPP to CCFG

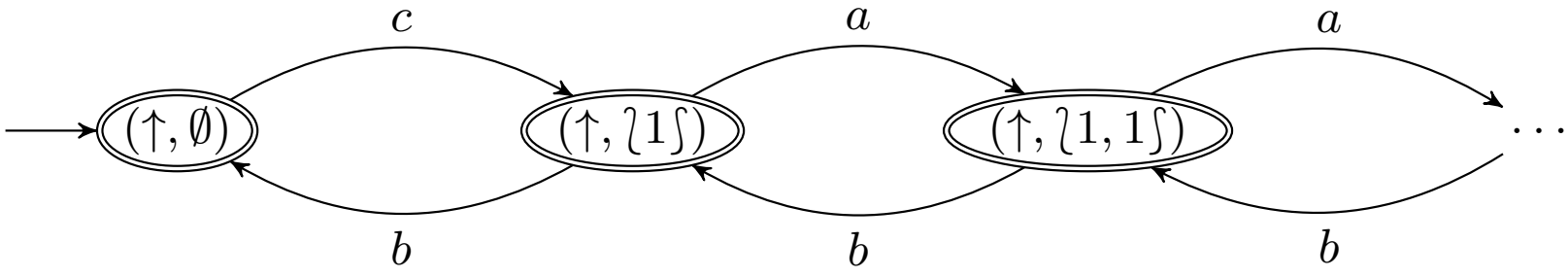
$$S \stackrel{\text{def}}{=} \mathbf{1} + a.(S \parallel B) + c.(S \parallel D) \qquad B \stackrel{\text{def}}{=} b.\mathbf{1} \qquad D \stackrel{\text{def}}{=} \mathbf{1} + d.\mathbf{1}$$



CWI

One-state counterexample





CWI

Specification of the bag.

$$AB \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} ins(d).(AB \parallel (\mathbf{1} + rem(d).\mathbf{1})) \ .$$

$$ABag \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} ins(d).\theta(ABag \parallel (\mathbf{1} + rem(d).\mathbf{1})) +$$

$$+ \sum_{d \in \mathcal{D}} show(^{-}d).ABag \ .$$

CWI

Priorities.

$$\frac{p\downarrow}{\theta(p)\downarrow} \qquad \frac{p \xrightarrow{a} p' \qquad \forall b > a \quad p \not\xrightarrow{b}}{\theta(p) \xrightarrow{a} \theta(p')}$$

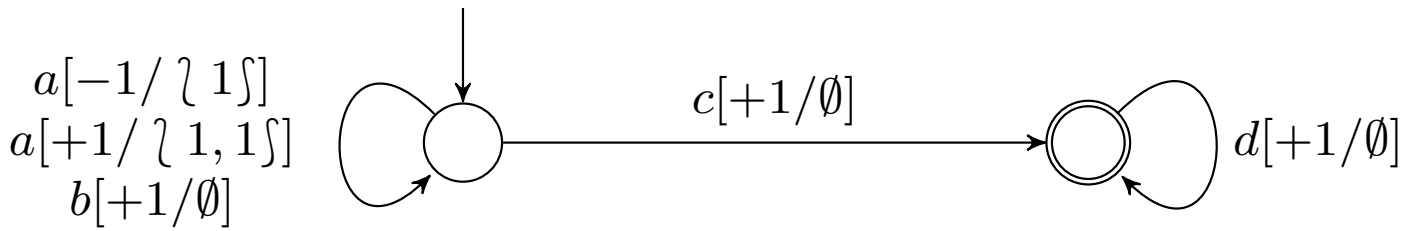
$$\frac{p \xrightarrow{a} p'}{\rho_f(p) \xrightarrow{f(a)} \rho_f(p')} \qquad \frac{p\downarrow}{\rho_f(p)\downarrow}$$

CWI

Theorem: For every one-state parallel pushdown automaton there is a finite weakly guarded $\mathrm{BPP}_{\theta}^{\mathbf{01}}$ specification such that their process graphs are divergence-preserving branching bisimilar.

CWI

Not for two states.



CWI

Basic Communicating Processes

$$\frac{p \xrightarrow{c!d} p' \qquad q \xrightarrow{c?d} q'}{p \parallel q \xrightarrow{c(d)} p' \parallel q' \quad q \parallel p \xrightarrow{c(d)} q' \parallel p'}$$

$$\frac{p\downarrow}{\partial_C(p)\downarrow} \qquad \frac{p \xrightarrow{a} p' \qquad a \notin COM_C}{\partial_C(p) \xrightarrow{a} \partial_C(p')}$$

$$\frac{p\downarrow}{\tau_C(p)\downarrow} \qquad \frac{p \xrightarrow{c(d)} p' \qquad c \in C}{\tau_C(p) \xrightarrow{\tau} \tau_C(p')} \qquad \frac{p \xrightarrow{a} p' \qquad a \neq c(d) \text{ for } c \in C}{\tau_C(p) \xrightarrow{a} \tau_C(p')}$$

CWI

Communicating Bags

$$ABag^{io} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} i?d.\theta(ABag^{io} \parallel (\mathbf{1} + o!(^{+}d).\mathbf{1})) + \sum_{d \in \mathcal{D}} o!(^{-}d).ABag^{io}$$

$$ABag^{io} \leftrightarrows_{\mathsf{b}}^{\Delta} \tau_{\{\ell\}}(\partial_{\{\ell\}}(ABag^{i\ell} \parallel ABag^{\ell o}))$$

CWI

Correspondence

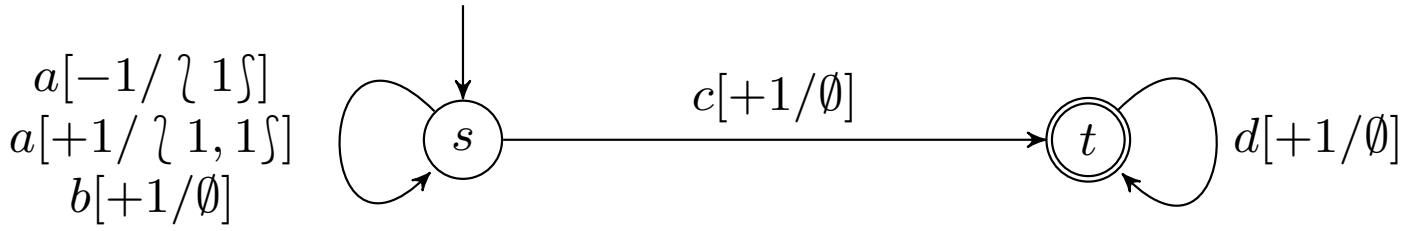
Theorem. For every parallel pushdown automaton there is a finite weakly guarded BCP-specification such that their process graphs are divergence-preserving branching bisimilar.

Proof. Again, we have parallel head normal form and Greibach normal form under $\leftrightarrows$

$$X \stackrel{\text{def}}{=} (\mathbf{1}+) \sum_{i=1}^{n} a_i.\tau_C(\partial_C(\theta(\xi_i))).$$

CWI

Two-state example



$$S \stackrel{\text{def}}{=} a^{-}.\tau_p(\partial_p(\theta(p!s.\mathbf{1} \parallel X_0 \parallel X_1)))$$

$$X_1 \stackrel{\text{def}}{=} p?s.((a^{+}.p!s.\mathbf{1} \parallel X_1 \parallel X_1)+(b.p!s.\mathbf{1})+(c.p!t.\mathbf{1}))+p?t.(\mathbf{1}+d.p!t.\mathbf{1})$$

$$X_0 \stackrel{\text{def}}{=} p?s.(a^{-}.p!s.\mathbf{1} \parallel X_1) + p?t.\mathbf{1}$$

Can do without priorities, but then we need a countable set of values and countable sum.

CWI

Communication not too powerful

For every finite weakly guarded BCP-specification there is a parallel pushdown automaton such that their process graphs are divergence-preserving rooted branching bisimilar.

CWI

A characterization

A process p is a parallel push-down process, if and only there is a regular process q such that

$$p \leftrightarrows^{\Delta}_{\mathsf{b}} \tau_{\{i,o\}}(\partial_{\{i,o\}}(q \parallel ABag^{io}))$$

Can do without priorities, but then we need *get* communication (asymmetric communication).

CWI

Conclusion

The set of processes given by a parallel pushdown automaton coincides with the set of processes given by a finite weakly guarded recursive specification over a process algebra with actions, choice, priorities and parallel composition with value passing communication.