Towards a proof analysis of processes

Cosimo Perini Brogi
IMT School for Advanced Studies Lucca

IT Matters Final Workshop
July 11-12, 2023
Lucca

General framework

Main goal

Apply proof-theoretic tools and techniques to formal verification of concurrent programs

General framework

Main goal

Apply proof-theoretic tools and techniques to formal verification of concurrent programs

How to reach it

- Exploit the interplay between (non-classical) logics, process algebras/calculi and labelled transition systems
- From the particular to the general, starting with experiments on some standard and well-understood process calculi

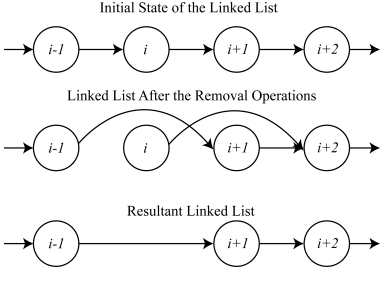
Desiderata

- If the verification goal is complex, break it into subgoals, according to its syntactic structure
  $\Rightarrow$ basic requirement for automation of proof search algorithms (Troelstra and Schwichtenberg 2000)
- If the process under investigation is complex, prove that it does satisfy a certain property by proving that its subprocesses do satisfy some properties that are sufficient to establish the original property
  $\Rightarrow$ verification of modular processes should be modular (Stirling 1987)

Outline

Concurrent processes

Example from informatics



A simple example of two processes modifying a linked list at the same time causing a conflict (WikiMedia, CC-BY-SA-3.0)

Concurrent processes

Example from biology

Starlings flocks, murmuration (WikiMedia, CC-BY-SA-2.0)

Concurrent processes

Example from epic (?!)

Orlando furioso, "*il poema dal movimento errante, a zig zag*" (I. Calvino)

Calculus of communicating systems

Key insight

Hoare's and Milner's proposal

Process calculi provide a syntactic characterisation of concurrent programs that is based on process operators building new process behaviours from simpler ones

To describe processes, focus on interactions!

Calculus of communicating systems

Syntax-driven SOS

$$\textsc{No rules for } \mathbf{0}$$

$$\textsc{Act}\ \frac{}{\mu.p \xrightarrow{\mu} p}$$

$$\textsc{Def}\ \frac{p \xrightarrow{\mu} q \qquad p \triangleq k}{k \xrightarrow{\mu} q}$$

$$\textsc{Ren}\ \frac{p \xrightarrow{\mu} q}{p[f] \xrightarrow{f(\mu)} q[f]}$$

$$\textsc{Res}\ \frac{p \xrightarrow{\mu} q}{p_{\setminus L} \xrightarrow{\mu} q_{\setminus L}}\ \mu, \overline{\mu} \notin L$$

$$\textsc{Sum}_1\ \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} p'}$$

$$\textsc{Sum}_2\ \frac{q \xrightarrow{\mu} q'}{p + q \xrightarrow{\mu} q'}$$

$$\textsc{Com}_1\ \frac{p \xrightarrow{\mu} p'}{p|q \xrightarrow{\mu} p'|q}$$

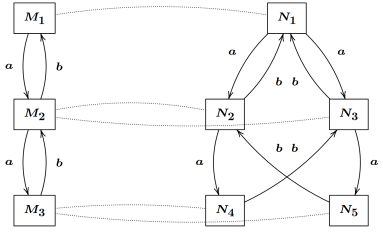
$$\textsc{Com}_2\ \frac{q \xrightarrow{\mu} q'}{p|q \xrightarrow{\mu} p|q'}$$

$$\textsc{Com}_3\ \frac{p \xrightarrow{\lambda} p' \qquad q \xrightarrow{\overline{\lambda}} q'}{p|q \xrightarrow{\tau} p'|q'}$$

Behavioural equivalence via structural semantics

Using the structural semantics, labelled transition systems are rigorously associated to concurrent processes:
Just consider $\mathcal{T} := \langle \mathcal{P}, \mathcal{A}_\tau, \rightarrow \rangle$, where $\rightarrow$ applied to μ is the least relation on $\mathcal{P}$ generated by the rules of the structural semantics.

When two CCS are observationally identifiable?



Two states are bisimilar (and we write $p \simeq q$) when there exists a bisimulation R such that pRq.

Quotienting LTS over $\simeq$ provides a semantics of CCS that focuses of (some) external behaviour of process, abstracting from their specific identity.

Behavioural equivalence via logic

Definition (Hennessy-Milner logic)

Formulas of HM are defined by the following grammar:

$$A \in \mathtt{Frm}_{\mathsf{HM}} ::= \top \mid \neg A \mid A \wedge B \mid \langle \mu \rangle A,$$

where $\mu \in \mathcal{A}_\tau$, $\neg$ and $\wedge$ denote classical negation and conjunction, resp.

Now, given the LTS $\langle \mathcal{P}, \mathcal{A}_\tau, \rightarrow \rangle$, we can define a standard notion of local forcing as follows:

- $p \Vdash \top$ for any $p \in \mathcal{P}$;
- $p \Vdash \neg A$ iff $p \nVdash A$;
- $p \Vdash A \wedge B$ iff $p \Vdash A$ and $p \Vdash B$;
- $p \Vdash \langle \mu \rangle A$ iff there exists a $q \in \mathcal{P}$ such that $p \xrightarrow{\mu} q$ and $q \Vdash A$.

Behavioural equivalence via logic

Theorem (Hennessy and Milner 1985)

Let's say that a state p of an LTS $\mathcal{T}$ is finitely branching if the set of states that are reachable in $\mathcal{T}$ from p is finite.
Then, given two finitely branching states p, q

$$p \simeq q \text{ iff, for any } A \in \mathtt{Frm}_{HM},\ p \Vdash A \text{ iff } q \Vdash A.^{a}$$

[a] The finite branching condition can be discarded if infinite conjunctions are allowed in the basic language.

Bits of proof theory

Sequents

A sequent is an expression $\Gamma \vdash \Delta$, where Γ and Δ are finite *multisets* of formulas in a given language.
A sequent $A_1, \cdots, A_n \vdash B_1, \cdots, B_m$ can be interpreted as

$$\bigwedge_{i=1}^{n} A_i \to \bigvee_{j=1}^{m} B_j,$$

where $\bigwedge \varnothing = \top$ and $\bigvee \varnothing = \bot$.

A sequent calculus is a set of rules with shape

$$\frac{\mathcal{S}_1 \qquad \cdots \qquad \mathcal{S}_n}{\mathcal{S}_0}\ \mathit{Rule}$$

for certain sequents $\mathcal{S}_0, \cdots, \mathcal{S}_n$.

Bits of proof theory

G3-style sequent calculi

For classical propositional logic, we have the following rules:

$$\frac{}{p, \Gamma \Rightarrow \Delta, p} \qquad \frac{}{\bot, \Gamma \Rightarrow \Delta}\, L\bot$$

$$\frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\, L\wedge \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}\, R\wedge$$

$$\frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta}\, L\vee \qquad \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}\, R\vee$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\, L\neg \qquad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\, R\neg$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad B, \Gamma \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\, L\to \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \to B}\, R\to$$

Bits of proof theory

Cut elimination

The rules for classical propositional logic are very well-designed, but in order to capture the standard mathematical reasoning we want the following rule to be admissible:

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \mathit{Cut}$$

A key result in any structural analysis of a logic is the following canonical form theorem:

Theorem (*Hauptsatz, Gentzen 1934*)

Any derivation of a sequent in the calculus for classical (propositional) logic can be effectively turned into a derivation that does not use the cut rule.

Bits of proof theory

Applications

Gentzen's proof has a direct application in the field of computerised reasoning.
In fact, theorem provers do work since they are based on logical calculi having good structural properties:

- Analyticity: Each formula occurring in a cut-free derivation is a subformula of the formulas occurring lower in the derivation branch
  $\leadsto$ *no guesses are required* to the prover when developing a formal proof;

Bits of proof theory

Applications

Gentzen's proof has a direct application in the field of computerised reasoning.
In fact, theorem provers do work since they are based on logical calculi having good structural properties:

- Analyticity: Each formula occurring in a cut-free derivation is a subformula of the formulas occurring lower in the derivation branch
  $\rightsquigarrow$ *no guesses are required* to the prover when developing a formal proof;
- Avoiding of backtracking: For each rule of a G3-system, derivability of the conclusion implies the derivability of the premise(s)
  $\rightsquigarrow$ *no backtracking* during the proof search and *no bit of information gets lost* during the procedure;

Bits of proof theory

Applications

Gentzen's proof has a direct application in the field of computerised reasoning.
In fact, theorem provers do work since they are based on logical calculi having good structural properties:

- Analyticity: Each formula occurring in a cut-free derivation is a subformula of the formulas occurring lower in the derivation branch
  $\leadsto$ *no guesses are required* to the prover when developing a formal proof;
- Avoiding of backtracking: For each rule of a G3-system, derivability of the conclusion implies the derivability of the premise(s)
  $\leadsto$ *no backtracking* during the proof search and *no bit of information gets lost* during the procedure;
- Termination: Each proof search must come to an end (in the propositional setting), because of the *subformula property*, or a specific proof search algorithm
  $\leadsto$ no loops of the prover

Principled prototype

Basic language

Our proof system G3CCS is based on the explicit internalisation in the sequents of the semantics for CCS.

Principled prototype

Basic language

Our proof system G3CCS is based on the explicit internalisation in the sequents of the semantics for CCS.
We work with labelled formulas with shape:

$$p \xrightarrow{\mu} q \quad | \quad p \triangleq k \quad | \quad p \equiv q \quad | \quad p : A$$

Principled prototype

Basic language

Our proof system G3CCS is based on the explicit internalisation in the sequents of the semantics for CCS.
We work with labelled formulas with shape:

$$p \xrightarrow{\mu} q \quad | \quad p \triangleq k \quad | \quad p \equiv q \quad | \quad p : A$$

Sequents are now expressions $\Gamma \vdash \Delta$ where Γ, Δ are finite multisets of labelled formulas, and in Δ only formulas with shape $p : A$ may occur.

Principled prototype

Logical rules

$$\frac{}{\Gamma \vdash \Delta, p : \top}\, R\top$$

$$\frac{\Gamma \vdash \Delta, p : A}{p : \neg A, \Gamma \vdash \Delta}\, L\neg \qquad \frac{p : A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, p : \neg A}\, R\neg$$

$$\frac{p : A, p : B, \Gamma \vdash \Delta}{p : A \wedge B, \Gamma \vdash \Delta}\, L\wedge \qquad \frac{\Gamma \vdash \Delta, p : A \qquad \Gamma \vdash \Delta, p : B}{\Gamma \vdash \Delta, p : A \wedge B}\, R\wedge$$

$$\frac{p \xrightarrow{\mu} y, y : A, \Gamma \vdash \Delta}{p : \langle \mu \rangle A, \Gamma \vdash \Delta}\, L\Diamond_{(!y)} \qquad \frac{p \xrightarrow{\mu} q, \Gamma \vdash \Delta, p : \langle \mu \rangle A, q : A}{p \xrightarrow{\mu} q, \Gamma \vdash \Delta, p : \langle \mu \rangle A}\, R\Diamond$$

Principled prototype

Compositional rules

For each process operator, we need to introduce in G3CCS some rules characterising it in terms of the SOS rules for itself.
Notice first that each of these rules can be translated into geometric formulas.[1]

After (Negri and von Plato 2011)

Use geometric sequent rules...

[1] A geometric formula is a formula in the language of first-order classical logic of shape $A \to B$, where A, B do not contain $\forall$ and $\to$.

Worked-out examples

Choice, top-down

$$\textsc{Sum}_1 \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} p'} \quad \rightsquigarrow \quad (p \xrightarrow{\mu} p') \to (p + q \xrightarrow{\mu} p')$$

$$\rightsquigarrow \quad \frac{p + q \xrightarrow{\mu} p', p \xrightarrow{\mu} p', \Gamma \vdash \Delta}{p \xrightarrow{\mu} p', \Gamma \vdash \Delta} \textsc{Sum}_{direct1}$$

Worked-out examples

Choice, bottom-up

$$\textsc{Sum}_1\ \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} p'} \quad + \quad \textsc{Sum}_2\ \frac{q \xrightarrow{\mu} q'}{p + q \xrightarrow{\mu} q'} \qquad \rightsquigarrow \qquad (p + q \xrightarrow{\mu} x) \to (p \xrightarrow{\mu} x) \vee (q \xrightarrow{\mu} x)$$

$$\rightsquigarrow \qquad \frac{p \xrightarrow{\mu} x, p + q \xrightarrow{\mu} x, \Gamma \vdash \Delta \qquad q \xrightarrow{\mu} x, p + q \xrightarrow{\mu} x, \Gamma \vdash \Delta}{p + q \xrightarrow{\mu} x, \Gamma \vdash \Delta}$$

Worked-out examples

Communication, top-down

$$\textsc{Com}_1 \frac{p \xrightarrow{\mu} p'}{p|q \xrightarrow{\mu} p'|q} \quad \rightsquigarrow \quad (p \xrightarrow{\mu} p') \to (p|q \xrightarrow{\mu} p'|q)$$

$$\rightsquigarrow \quad \frac{p|q \xrightarrow{\mu} p'|q, p \xrightarrow{\mu} p', \Gamma \vdash \Delta}{p \xrightarrow{\mu} p', \Gamma \vdash \Delta} \textsc{Com}_{\mathit{direct1}}$$

Worked-out examples

Communication, top-down

$$\textsc{Com}_3 \frac{p \xrightarrow{\lambda} p' \qquad q \xrightarrow{\overline{\lambda}} q'}{p|q \xrightarrow{\tau} p'|q'} \quad \rightsquigarrow \quad (p \xrightarrow{\lambda} p') \wedge (q \xrightarrow{\overline{\lambda}} q') \rightarrow (p|q \xrightarrow{\tau} p'|q')$$

$$\rightsquigarrow \quad \frac{p|q \xrightarrow{\tau} p'|q', p \xrightarrow{\lambda} p', q \xrightarrow{\overline{\lambda}} q', \Gamma \vdash \Delta}{p \xrightarrow{\lambda} p', q \xrightarrow{\overline{\lambda}} q', \Gamma \vdash \Delta} \textsc{Com}_{direct3}$$

Worked-out examples

Communication, bottom-up

$$\textsc{Com}_1 + \textsc{Com}_2 + \textsc{Com}_3 \rightsquigarrow (p|q \xrightarrow{\mu} z) \to (\exists x, p \xrightarrow{\mu} x \wedge z \equiv x|q) \vee (\exists y, q \xrightarrow{\mu} y \wedge z \equiv p|y)$$

$$\& \quad (p|q \xrightarrow{\tau} z) \to (\exists x, p \xrightarrow{\tau} x \wedge z \equiv x|q) \vee (\exists y, q \xrightarrow{\tau} y \wedge z \equiv p|y) \vee (\exists x \exists y, p \xrightarrow{\lambda} x \wedge q \xrightarrow{\overline{\lambda}} y \wedge z \equiv x|y)$$

$$\rightsquigarrow \quad \frac{x|q \equiv z, p \xrightarrow{\mu} x, p|q \xrightarrow{\mu} z, \Gamma \vdash \Delta \qquad p|y \equiv z, q \xrightarrow{\mu} y, p|q \xrightarrow{\mu} z, \Gamma \vdash \Delta}{p|q \xrightarrow{\mu} z, \Gamma \vdash \Delta}\ \textsc{Com}_{\mathit{var1}(!x,!y)}$$

$$\& \quad \frac{x|q \equiv z, p \xrightarrow{\tau} x, p|q \xrightarrow{\tau} z, \Gamma \vdash \Delta \qquad p|y \equiv z, q \xrightarrow{\tau} y, p|q \xrightarrow{\tau} z, \Gamma \vdash \Delta \qquad x|y \equiv z, p \xrightarrow{\lambda} x, q \xrightarrow{\overline{\lambda}} y, p|q \xrightarrow{\tau} z, \Gamma \vdash \Delta}{p|q \xrightarrow{\tau} z, \Gamma \vdash \Delta}\ \textsc{Com}_{\mathit{var2}(!x,!y)}$$

Main results

Theorem (Structural completeness)

G3CCS *satisfies the following properties:*

- *Generalised initial sequents are derivable;*
- *Substitution rule for states over variables are height-preserving admissible;*
- *Weakening rules are height preserving admissible;*
- *All the rules are height-preserving invertible;*
- *Contraction rules are height-preserving admissible;*
- *The cut rule can be effectively eliminated.*

If we have a derivation of a parametrised property $x_1 : A_1, \cdots, x_n : A_n \vdash op(x_1, \cdots, x_n) : B$ (where op is process operator) and derivations of $\vdash p_1 : A_1, \cdots, \vdash p_n : A_n$, then we can apply substitution and cut to obtain a derivation of $\vdash op(p_1, \cdots, p_n) : B$

$\Rightarrow$ Compositional verification!

Main results

Theorem (Semantic completeness)

G3CCS *satisfies the following properties:*

Soundness: *If the sequent* $\Gamma \vdash \Delta$ *is derivable, then* $\Gamma \vDash \Delta$*;*

Completeness: *If the sequent* $\Gamma \vdash \Delta$ *is not derivable, then it is possible to extract from the failed proof search an LTS-countermodel to* $\Gamma \vdash \Delta$ *(complications to be checked)*

Put in perspective

- It is possible to use the most important tools and techniques of proof theory in formal methods for concurrent processes described by CCS (and other process calculi)
- Contemporary research in internalisation of mathematical structures in sequent calculi finds a natural application in a principled verification (no black boxes) of reactive systems
- The theory needs some improvement (termination of backward proof search algorithm?), with the aim of implementing this kind of calculi (automation in Prolog?)

Put in perspective

- It is possible to use the most important tools and techniques of proof theory in formal methods for concurrent processes described by CCS (and other process calculi)
- Contemporary research in internalisation of mathematical structures in sequent calculi finds a natural application in a principled verification (no black boxes) of reactive systems
- The theory needs some improvement (termination of backward proof search algorithm?), with the aim of implementing this kind of calculi (automation in Prolog?)

Many thanks for listening!

References

- De Nicola, R. (2014). A gentle introduction to Process Algebras. Notes, 7.
- Hennessy, M., Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. Journal of the ACM (JACM), 32(1), 137-161.
- Negri, S., von Plato, J. (2011). Proof analysis: a contribution to Hilbert's last problem. Cambridge University Press.
- Stirling, C. (1987). Modal logics for communicating systems. Theoretical Computer Science, 49(2-3), 311-347.
- Troelstra, A. S., Schwichtenberg, H. (2000). Basic proof theory (No. 43). Cambridge University Press.