Unique Solutions of Contractions, CCS, and their HOL Formalisation

Chun Tian [1]   Davide Sangiorgi [2]

[1]Fondazione Bruno Kessler (FBK), Italy

[2]Università di Bologna, Italy and INRIA, France

September 3, 2018

Project Motivation

- Concurrency Theory is important for understanding concurrent and reactive systems;
- Milner's Calculus of Communicating Systems (CCS) is simple, elegant process calculi widely adopted in Concurrency Theory courses, yet textbooks cannot provide all proof details;
- The CCS formalisation project is a good chance for learning Interactive Theorm Proving (ITP), with minimal dependencies on other formal theories.

Project summary

- 20,000 lines of Standard ML code;
- 500 *manually* proved lemmas/theorems.

Availabile in HOL official examples: https://github.com/HOL-Theorem-Prover/HOL/tree/master/examples/CCS

Calculus of Communicating Systems (CCS)

Definition (Actions and CCS processes)

$$
\begin{array}{lll}
\mu & := & \tau \;\big|\; a \;\big|\; \overline{a} \\
P & := & \mathbf{0} \;\big|\; \mu.\,P \;\big|\; P_1 \mid P_2 \;\big|\; P_1 + P_2 \;\big|\; (\boldsymbol{\nu} a)\,P \;\big|\; P\,[rf] \;\big|\; A \;\big|\; \texttt{rec}\,A.\,P
\end{array}
$$

Definition (Structural Operational Semantics)

$$
\frac{}{\mu.\,P \xrightarrow{\mu} P}
\qquad
\frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'}
\qquad
\frac{Q \xrightarrow{\mu} Q'}{P + Q \xrightarrow{\mu} Q'}
\qquad
\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}
$$

$$
\frac{Q \xrightarrow{\mu} Q'}{P \mid Q \xrightarrow{\mu} P \mid Q'}
\qquad
\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}
\qquad
\frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\,P \xrightarrow{\mu} (\boldsymbol{\nu} a)\,P'}\; \mu \neq a, \overline{a}
$$

$$
\frac{P \xrightarrow{\mu} P'}{P\,[rf] \xrightarrow{rf(\mu)} P'\,[rf]}\; \forall a.\; rf(\overline{a}) = \overline{rf(a)}
\qquad
\frac{P\{\texttt{rec}\,A.\,P/A\} \xrightarrow{\mu} P'}{\texttt{rec}\,A.\,P \xrightarrow{\mu} P'}
$$

Bisimulation and Bisimiarity

Definition

A process relation $\mathcal{R}$ is a *strong bisimulation* if, whenever $P \mathcal{R} Q$, we have:

1. $P \xrightarrow{\mu} P'$ implies that there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$;
2. $Q \xrightarrow{\mu} Q'$,implies that there is P' such that $P \xrightarrow{\mu} P'$ and $P' \mathcal{R} Q'$.

P and Q are *bisimilar* ($P \sim Q$), if $P \mathcal{R} Q$ for some bisimulation $\mathcal{R}$.

Definition

A process relation $\mathcal{R}$ is a *weak bisimulation* if, whenever $P \mathcal{R} Q$, we have:

1. $P \xrightarrow{\mu} P'$ implies that there is Q' such that $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$
2. $Q \xrightarrow{\mu} Q'$,implies that there is P' such that $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ and $P' \mathcal{R} Q'$.

P and Q are *weakly bisimilar* ($P \approx Q$), if $P \mathcal{R} Q$ for some bisimulation $\mathcal{R}$.

Rooted Bisimiarity (Observation Congruence)

Definition

Two processes P and Q are **rooted bisimilar** ($P \approx^{c} Q$), if we have:

1. $P \xrightarrow{\mu} P'$ implies that there is Q' such that $Q \stackrel{\mu}{\Longrightarrow} Q'$ and $P' \approx Q'$;
2. $Q \xrightarrow{\mu} Q'$ implies that there is P' such that $P \stackrel{\mu}{\Longrightarrow} P'$ and $P' \approx Q'$.

Theorem

1. $P \approx^{c} Q \Rightarrow P \approx Q$;
2. $\sim$ *and* $\approx^{c}$ *are preserved by all CCS operators;* *(Congruence)*
3. $\approx$ *is preserved by all CCS operators but direct sums;* *("Congruence")*
4. $P \approx Q \;\Leftrightarrow\; (P \approx^{c} Q \vee P \approx^{c} \tau.\, Q \vee \tau.\, P \approx^{c} Q)$. *(Hennessy Lemma)*
5. $P \approx^{c} Q \;\Leftrightarrow\; (\forall R.\;\, P + R \approx Q + R)$. *(Coarsest Congruence in $\approx$)*

Unique Solution of Equations (Robin Milner, 1989)

Theorem (for $\sim$)

Let $\widetilde{E}$ be weakly guarded with free variables at most $\widetilde{X}$, and let $\widetilde{P} \sim \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \sim \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, then $\widetilde{P} \sim \widetilde{Q}$.

Theorem (for $\approx$, not explicitly appeared)

Let $\widetilde{E}$ be guarded and sequential with only guarded sums and free variables at most $\widetilde{X}$, and let $\widetilde{P} \approx \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \approx \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, then $\widetilde{P} \approx \widetilde{Q}$.

Theorem (for $\approx^{\mathrm{c}}$)

Let $\widetilde{E}$ be guarded and sequential with free variables at most $\widetilde{X}$, and let $\widetilde{P} \approx^{\mathrm{c}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \approx^{\mathrm{c}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, then $\widetilde{P} \approx^{\mathrm{c}} \widetilde{Q}$.

Conditions required by Milner's theorems (for $\sim$, $\approx^{\mathrm{c}}$)

Definition

X is *weakly guarded* in E if each occurrence of X is within some subexpression $\mu.\,F$ of E.

Definition

X is *(strongly) guarded* in E if each occurrence of X is within some subexpression $l.\,F$ of E.
X is *sequential* in E if every subexpression of E which contains X, apart from X itself, is of the form $\mu.\,F$ or $\Sigma\widetilde{F}$.

1. Any P (without X) is guarded and sequential;
2. If E is sequential, then $l.\,E$ is guarded and sequential;
3. If E is guarded and sequential, so is $\mu.\,E$;
4. If E_1 and E_2 are both guarded and sequential, so is $E_1 + E_2$.

Conditions required by Milner's theorem (for $\approx$)

Definition

X is *sequential with only guarded sums* in E if every subexpression of E which contains X, apart from X itself, is of the form $\mu.\, F$ or $\Sigma\mu_i.\, F_i$.

1. Any P (without X) is guarded and "sequential";
2. If E is "sequential", then $l.\, E$ is guarded and "sequential";
3. If E is guarded and "sequential", so is $\mu.\, E$;
4. If E_1 and E_2 are guarded and "sequential", so is $\tau.\, E_1 + \tau.\, E_2$;
5. If E_1 is guarded and "sequential", E_2 is "sequential", then $\tau.\, E_1 + l.\, E_2$ is guarded and "sequential";
6. If E_1 is "sequential", E_2 is guarded and "sequential", then $l.\, E_1 + \tau.\, E_2$ is guarded and "sequential";
7. If E_1 and E_2 are "sequential", $l_1.\, E_1 + l_2.\, E_2$ is guarded and "sequential".

A refinement of Milner's technique (D. Sangiorgi, 2015)

Definition (Contraction)

A process relation $\mathcal{R}$ is a *(bisimulation) contraction* if whenever $P \mathcal{R} Q$,

1. $P \xrightarrow{\mu} P'$ implies there is Q' such that $Q \xrightarrow{\widehat{\mu}} Q'$ and $P' \mathcal{R} Q'$;
2. $Q \xrightarrow{\mu} Q'$ implies there is P' such that $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ and $P' \approx Q'$.

Bisimilarity contraction, written as $P \succeq_{\mathrm{bis}} Q$, if $P \mathcal{R} Q$ for some contraction $\mathcal{R}$.

Lemma (Precongruence of $\succeq_{\mathrm{bis}}$ in CCS)

$\succeq_{\mathrm{bis}}$ *is a preorder (reflexive, transitive) and is preserved by all CCS operators but direct sums.*

Theorem (Unique Solution of Contractions)

Let $\widetilde{E}$ *be weakly guarded with only guarded sums and free variables at most* $\widetilde{X}$, *and let* $\widetilde{P} \succeq_{\mathrm{bis}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \succeq_{\mathrm{bis}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, *then* $\widetilde{P} \approx \widetilde{Q}$.

Further refinements (C. Tian, 2017; the current paper)

Definition (Rooted contraction)

Two processes P and Q are in *rooted contraction*, written as $P \succeq^{c}_{\mathrm{bis}} Q$, if

1. $P \xrightarrow{\mu} P'$ implies that there is Q' with $Q \xrightarrow{\mu} Q'$ and $P' \succeq_{\mathrm{bis}} Q'$;
2. $Q \xrightarrow{\mu} Q'$ implies that there is P' with $P \overset{\mu}{\Longrightarrow} P'$ and $P' \approx Q'$.

Lemma (Precongruence of $\succeq^{c}_{\mathrm{bis}}$ in CCS)

$\succeq_{\mathrm{bis}}$ *is a preorder (reflexive, transitive) and is preserved by all CCS operators.*

Theorem (Unique Solution of Rooted Contractions)

Let $\widetilde{E}$ be weakly guarded with free variables at most $\widetilde{X}$, and let $\widetilde{P} \succeq^{c}_{\mathrm{bis}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \succeq^{c}_{\mathrm{bis}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, then $\widetilde{P} \approx^{c} \widetilde{Q}$ (thus also $\widetilde{P} \approx \widetilde{Q}$).

CCS Formalisation in HOL (Monica Nesi and Chun Tian)

Name	**Summary**	**Lines**
CCSTheory	Basic CCS definitions, SOS rules	1009
CCSConv	Decision procedure of CCS transitions	562
StrongEQTheory	Strong bisimulation and bisimilarity ($\sim$)	634
StrongLawsTheory	Algebraic laws of $\sim$; expansion law	2002
WeakEQTheory	Weak bisimulation and bisimilarity	1974
WeakLawsTheory	Algebraic laws of $\approx$	335
ObsCongrTheory	Observation congruence ($\approx^c$)	697
ObsCongrLawsTheory	Algebraic laws of $\approx^c$	402
BisimulationUptoTheory	Bisimulation up to $\sim$, $\approx$ and $\approx^c$	1180
CongruenceTheory	Context, guardedness and congruence	1457
CoarsestCongrTheory	Deep results between $\approx$ and $\approx^c$	872
TraceTheory	Trace and relationship with weak transition	753
ExpansionTheory	Expansion preorder, precongruence	976
ContractionTheory	Contraction preorder, precongruence	2068
UniqueSolutionTheory	Unique solution of equations/contractions	2386

Higher Order Logic (HOL)

Definition (Type in HOL)

$$\sigma \quad ::= \quad \alpha \quad | \quad c \quad | \quad \underbrace{(\sigma_1, \ldots, \sigma_n) op} \quad | \quad \underbrace{\sigma_1 \to \sigma_2}$$

- α: type variables
- c: atomic types
- $(\sigma_1, \ldots, \sigma_n) op$: compound types
- $\sigma_1 \to \sigma_2$: function types (domain, codomain)

Definition (Term in HOL)

$$t \quad ::= \quad x \quad | \quad c \quad | \quad \underbrace{t\ t'} \quad | \quad \underbrace{\lambda x.\ t}$$

- x: variables
- c: constants
- $t\ t'$: function applications (function t, argument t')
- $\lambda x.\ t$: λ-abstractions

Primitive rules

1. Assumption introduction [ASSUME],
2. Reflexivity [REFL],
3. β-conversion [BETA_CONV],
4. Substitution [SUBST],
5. Abstraction [ABS],
6. Type instantiation [INST_TYPE],
7. Discharging an assumption [DISCH],
8. Modus Ponens [MP]

Logical constants

$$\begin{array}{l}
\vdash \ \mathrm{T} = ((\lambda x_{\textbf{bool}}.\ x) = (\lambda x_{\textbf{bool}}.\ x)) \\
\vdash \ \forall = \lambda P_{\alpha \to \textbf{bool}}.\quad P = (\lambda x.\ \mathrm{T}) \\
\vdash \ \exists = \lambda P_{\alpha \to \textbf{bool}}.\quad P(\varepsilon\ P) \\
\vdash \ \mathrm{F} = \forall b_{\textbf{bool}}.\quad b \\
\vdash \ \neg = \lambda b.\quad b \Rightarrow \mathrm{F} \\
\vdash \ \wedge = \lambda b_1\ b_2.\ \forall b.\ (b_1 \Rightarrow (b_2 \Rightarrow b)) \Rightarrow b \\
\vdash \ \vee = \lambda b_1\ b_2.\ \forall b.\ (b_1 \Rightarrow b) \Rightarrow ((b_2 \Rightarrow b) \Rightarrow b)
\end{array}$$

Axioms

BOOL_CASES_AX	$\vdash \forall b.\ (b = \mathrm{T}) \vee (b = \mathrm{F})$
ETA_AX	$\vdash \forall f_{\alpha \to \beta}.\ (\lambda x.\ f\ x) = f$
SELECT_AX	$\vdash \forall P_{\alpha \to \textbf{bool}}\ x.\ P\ x \Rightarrow P(\varepsilon\ P)$
INFINITY_AX	$\vdash \exists f_{\textit{ind} \to \textit{ind}}.\ \mathrm{One\_One}\ f\ \wedge\ \neg(\mathrm{Onto}\ f)$

Syntax of CCS operators, constant and actions

New types: "`(α, β) CCS`", "`β Action`", "`β Label`" and "`β Relabeling`".

Operator	CCS Notation	HOL term	HOL (abbrev.)
nil	$\mathbf{0}$	`nil`	`nil`
prefix	$u.\,P$	`prefix u P`	u`..`P
sum	$P + Q$	`sum P Q`	$P + Q$
parallel	$P \mid Q$	`par P Q`	$P \parallel Q$
restriction	$(\nu\ L)\ P$	`restr L P`	$\nu\ L\ P$
recursion	`rec` $A.\,P$	`rec A P`	`rec` $A\ P$
relabeling	$P\ [rf]$	`relab P rf`	`relab` $P\ rf$
constant	A	`var A`	`var` A
invisible action	τ	`tau`	τ
input action	a	`label (name a)`	`In` a
output action	$\bar{a}$	`label (coname a)`	`Out` a

CCS transitions: an inductive relation

3-ary inductive relation `TRANS`: "`TRANS P u Q`" or "$P \ -u\rightarrow Q$".

$$\vdash u..P \ -u\rightarrow \ P \qquad \texttt{[PREFIX]}$$
$$\vdash P \ -u\rightarrow \ P' \Rightarrow P + Q \ -u\rightarrow \ P' \qquad \texttt{[SUM1]}$$
$$\vdash P \ -u\rightarrow \ P' \Rightarrow Q + P \ -u\rightarrow \ P' \qquad \texttt{[SUM2]}$$
$$\vdash P \ -u\rightarrow \ P' \Rightarrow P \parallel Q \ -u\rightarrow \ P' \parallel Q \qquad \texttt{[PAR1]}$$
$$\vdash P \ -u\rightarrow \ P' \Rightarrow Q \parallel P \ -u\rightarrow \ Q \parallel P' \qquad \texttt{[PAR2]}$$

$$\vdash P \ -\texttt{label}\ l\rightarrow \ P' \wedge Q \ -\texttt{label}\ (\texttt{COMPL}\ l)\rightarrow \ Q' \Rightarrow P \parallel Q \ -\tau\rightarrow \ P' \parallel Q' \qquad \texttt{[PAR3]}$$

$$\vdash P \ -u\rightarrow \ Q \wedge ((u = \tau) \vee (u = \texttt{label}\ l) \wedge l \notin L \wedge \texttt{COMPL}\ l \notin L) \Rightarrow \\ \nu\ L\ P \ -u\rightarrow \ \nu\ L\ Q \qquad \texttt{[RESTR]}$$

$$\vdash P \ -u\rightarrow \ Q \Rightarrow \texttt{relab}\ P\ rf \ -\texttt{relabel}\ rf\ u\rightarrow \ \texttt{relab}\ Q\ rf \qquad \texttt{[RELABELING]}$$

$$\vdash \texttt{CCS\_Subst}\ P\ (\texttt{rec}\ A\ P)\ A \ -u\rightarrow \ P' \Rightarrow \texttt{rec}\ A\ P \ -u\rightarrow \ P' \qquad \texttt{[REC]}$$

Relabeling and Substution

$\texttt{Is\_Relabeling}\ (f\ :\beta\ \texttt{Label} \to \beta\ \texttt{Label}) \iff$
$\forall (s\ :\beta).\ f\ (\texttt{coname}\ s) = \texttt{COMPL}\ (f\ (\texttt{name}\ s))$

$\texttt{relabel}\ rf\ \tau = \tau$
$\texttt{relabel}\ rf\ (\texttt{label}\ l) = \texttt{label}\ (\texttt{REP\_Relabeling}\ rf\ l)$

$\texttt{CCS\_Subst nil}\ E'\ X = \texttt{nil}$
$\texttt{CCS\_Subst}\ (u..E)\ E'\ X = u..\texttt{CCS\_Subst}\ E\ E'\ X$
$\texttt{CCS\_Subst}\ (E_1 + E_2)\ E'\ X =$
$\texttt{CCS\_Subst}\ E_1\ E'\ X + \texttt{CCS\_Subst}\ E_2\ E'\ X$
$\texttt{CCS\_Subst}\ (E_1 \parallel E_2)\ E'\ X =$
$\texttt{CCS\_Subst}\ E_1\ E'\ X \parallel \texttt{CCS\_Subst}\ E_2\ E'\ X$
$\texttt{CCS\_Subst}\ (\nu\ L\ E)\ E'\ X = \nu\ L\ (\texttt{CCS\_Subst}\ E\ E'\ X)$
$\texttt{CCS\_Subst}\ (\texttt{relab}\ E\ f)\ E'\ X = \texttt{relab}\ (\texttt{CCS\_Subst}\ E\ E'\ X)\ f$
$\texttt{CCS\_Subst}\ (\texttt{var}\ Y)\ E'\ X = \textbf{if}\ Y = X\ \textbf{then}\ E'\ \textbf{else}\ \texttt{var}\ Y$
$\texttt{CCS\_Subst}\ (\texttt{rec}\ Y\ E)\ E'\ X =$
$\textbf{if}\ Y = X\ \textbf{then}\ \texttt{rec}\ Y\ E$
$\textbf{else}\ \texttt{rec}\ Y\ (\texttt{CCS\_Subst}\ E\ E'\ X)$ [CCS_Subst_def]

Bisimulation and Bisimilarity

Definition

1. $E \overset{\epsilon}{\Rightarrow} E'$ (EPS E E'), EPS = $(\lambda E\ E'.\ E\ -\tau\rightarrow\ E')^*$
2. $E\ =u\Rightarrow\ E' \iff \exists E_1\ E_2.\ E \overset{\epsilon}{\Rightarrow} E_1 \wedge E_1\ -u\rightarrow\ E_2 \wedge E_2 \overset{\epsilon}{\Rightarrow} E'$
3. $\texttt{WEAK\_BISIM}\ \mathit{Wbsm} \iff$
$$
\begin{array}{l}
\forall E\ E'.\\
\quad \mathit{Wbsm}\ E\ E' \Rightarrow\\
\quad (\forall l.\\
\qquad (\forall E_1.\\
\qquad\quad E\ -\texttt{label}\ l\rightarrow\ E_1 \Rightarrow\\
\qquad\quad \exists E_2.\ E'\ =\texttt{label}\ l\Rightarrow\ E_2 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge\\
\qquad \forall E_2.\\
\qquad\quad E'\ -\texttt{label}\ l\rightarrow\ E_2 \Rightarrow \exists E_1.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge\\
\quad (\forall E_1.\ E\ -\tau\rightarrow\ E_1 \Rightarrow \exists E_2.\ E' \overset{\epsilon}{\Rightarrow} E_2 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge\\
\quad \forall E_2.\ E'\ -\tau\rightarrow\ E_2 \Rightarrow \exists E_1.\ E \overset{\epsilon}{\Rightarrow} E_1 \wedge \mathit{Wbsm}\ E_1\ E_2
\end{array}
$$
4. $E \approx E' \iff \exists \mathit{Wbsm}.\ \mathit{Wbsm}\ E\ E' \wedge \texttt{WEAK\_BISIM}\ \mathit{Wbsm}$

Bisimilarity as a fixed point

The actual definition of $\approx$ is automatically built by HOL4's `Hol_coreln`, coinducitve relation package:

```
val (WEAK_EQUIV_rules, WEAK_EQUIV_coind, WEAK_EQUIV_cases)
  = Hol_coreln `
  !E E'.
   (!l.
     (!E1. TRANS E  (label l) E1 ==>
           ?E2. WEAK_TRANS E' (label l) E2 /\ WEAK_EQUIV E1 E2) /\
     (!E2. TRANS E' (label l) E2 ==>
           ?E1. WEAK_TRANS E  (label l) E1 /\ WEAK_EQUIV E1 E2)) /\
   (!E1. TRANS E  tau E1 ==> ?E2. EPS E' E2 /\ WEAK_EQUIV E1 E2) /\
   (!E2. TRANS E' tau E2 ==> ?E1. EPS E  E1 /\ WEAK_EQUIV E1 E2)
   ==> WEAK_EQUIV E E' `;
```

`Hol_coreln` returns 3 theorems:

1. `WEAK_EQUIV_rule`: input "rules" proved as a theorem.
2. `WEAK_EQUIV_coind`: the resulting relation is maximal.
3. `WEAK_EQUIV_cases`: the resulting relation is a fix point.

Multi-hole contexts: inductive unary relation

Definition

$\texttt{CONTEXT}\ (\lambda t.\ t)$
$\texttt{CONTEXT}\ (\lambda t.\ p)$
$\texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ a..e\ t)$
$\texttt{CONTEXT}\ e_1 \wedge \texttt{CONTEXT}\ e_2 \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ e_1\ t + e_2\ t)$
$\texttt{CONTEXT}\ e_1 \wedge \texttt{CONTEXT}\ e_2 \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ e_1\ t \parallel e_2\ t)$
$\texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ \nu\ L\ (e\ t))$
$\texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf)$ [CONTEXT_rules]

The composition of two contexts is still a context:
$\vdash \texttt{CONTEXT}\ c_1 \wedge \texttt{CONTEXT}\ c_2 \Rightarrow \texttt{CONTEXT}\ (c_1 \circ c_2)$ [CONTEXT_combin]

Examples

$E[X] = a.X + b.X$ is presented as "$\lambda t.\ \ a..t + b..t$" $(a.[] + b.[])$).

Multi-hole contexts with only direct sums

Definition

$\texttt{GCONTEXT}\ (\lambda\, t.\ t)$
$\texttt{GCONTEXT}\ (\lambda\, t.\ p)$
$\texttt{GCONTEXT}\ e \Rightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ a..e\ t)$
$\texttt{GCONTEXT}\ e_1 \wedge \texttt{GCONTEXT}\ e_2 \Rightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ a_1..e_1\ t + a_2..e_2\ t)$
$\texttt{GCONTEXT}\ e_1 \wedge \texttt{GCONTEXT}\ e_2 \Rightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ e_1\ t \parallel e_2\ t)$
$\texttt{GCONTEXT}\ e \Rightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ \nu\ L\ (e\ t))$
$\texttt{GCONTEXT}\ e \Rightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ \texttt{relab}\ (e\ t)\ rf)$ [GCONTEXT_rules]

(GCONTEXT can be also seen as a normal context under special CCS syntax with only guarded sums $\Sigma\mu_i.\, p_i$.)

Congruence and precongruence

$\vdash$ `PreOrder` $R \iff$ `reflexive` $R \land$ `transitive` R
$\vdash$ `equivalence` $R \iff$ `reflexive` $R \land$ `symmetric` $R \land$ `transitive` R

$\vdash$ `precongruence` $R \iff$
`PreOrder` $R \land$
$\forall x\ y\ ctx.$ `CONTEXT` $ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$
$\vdash$ `precongruence1` $R \iff$
`PreOrder` $R \land$
$\forall x\ y\ ctx.$ `GCONTEXT` $ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$
$\vdash$ `congruence` $R \iff$
`equivalence` $R \land$
$\forall x\ y\ ctx.$ `CONTEXT` $ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$
$\vdash$ `congruence1` $R \iff$
`equivalence` $R \land$
$\forall x\ y\ ctx.$ `GCONTEXT` $ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$

$\vdash$ `congruence STRONG_EQUIV`
$\vdash$ `congruence1 WEAK_EQUIV`
$\vdash$ `congruence OBS_CONGR`

Weakly-guarded contexts: with and without direct sums

$$\begin{array}{l}\texttt{WG}\ (\lambda t.\ p)\\ \texttt{CONTEXT}\ e\ \Rightarrow\ \texttt{WG}\ (\lambda t.\ a..e\ t)\\ \texttt{WG}\ e_1\ \wedge\ \texttt{WG}\ e_2\ \Rightarrow\ \texttt{WG}\ (\lambda t.\ e_1\ t\ +\ e_2\ t)\\ \texttt{WG}\ e_1\ \wedge\ \texttt{WG}\ e_2\ \Rightarrow\ \texttt{WG}\ (\lambda t.\ e_1\ t\ \|\ e_2\ t)\\ \texttt{WG}\ e\ \Rightarrow\ \texttt{WG}\ (\lambda t.\ \nu\ L\ (e\ t))\\ \texttt{WG}\ e\ \Rightarrow\ \texttt{WG}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf) \qquad [\texttt{WG\_rules}]\end{array}$$

$$\begin{array}{l}\texttt{WGS}\ (\lambda t.\ p)\\ \texttt{GCONTEXT}\ e\ \Rightarrow\ \texttt{WGS}\ (\lambda t.\ a..e\ t)\\ \texttt{GCONTEXT}\ e_1\ \wedge\ \texttt{GCONTEXT}\ e_2\ \Rightarrow\ \texttt{WGS}\ (\lambda t.\ a_1..e_1\ t\ +\ a_2..e_2\ t)\\ \texttt{WGS}\ e_1\ \wedge\ \texttt{WGS}\ e_2\ \Rightarrow\ \texttt{WGS}\ (\lambda t.\ e_1\ t\ \|\ e_2\ t)\\ \texttt{WGS}\ e\ \Rightarrow\ \texttt{WGS}\ (\lambda t.\ \nu\ L\ (e\ t))\\ \texttt{WGS}\ e\ \Rightarrow\ \texttt{WGS}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf) \qquad [\texttt{WGS\_rules}]\end{array}$$

(Strongly) guarded contexts

$$
\begin{array}{l}
\mathtt{SG}\ (\lambda\, t.\ p) \\
\mathtt{CONTEXT}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \mathtt{label}\ l..e\ t) \\
\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ a..e\ t) \\
\mathtt{SG}\ e_1 \wedge \mathtt{SG}\ e_2 \Rightarrow \mathtt{SG}\ (\lambda\, t.\ e_1\ t + e_2\ t) \\
\mathtt{SG}\ e_1 \wedge \mathtt{SG}\ e_2 \Rightarrow \mathtt{SG}\ (\lambda\, t.\ e_1\ t \parallel e_2\ t) \\
\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \nu\ L\ (e\ t)) \\
\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \mathtt{relab}\ (e\ t)\ rf)
\end{array}
\qquad \mathtt{[SG\_rules]}
$$

There's no need to define special version of SG with only guarded sums, as "guarded and sequential" always appears together (but a single relation defnition is too complex).

Sequential contexts: with and without direct sums

"X is *sequential* in E if every subexpression of E which contains X, apart from X itself, is of the form $\mu. F$ or $\Sigma\widetilde{F}$."

$\texttt{SEQ}\ (\lambda t.\ t)$
$\texttt{SEQ}\ (\lambda t.\ p)$
$\texttt{SEQ}\ e \Rightarrow \texttt{SEQ}\ (\lambda t.\ a..e\ t)$
$\texttt{SEQ}\ e_1 \wedge \texttt{SEQ}\ e_2 \Rightarrow \texttt{SEQ}\ (\lambda t.\ e_1\ t + e_2\ t)$ [SEQ_rules]

$\texttt{GSEQ}\ (\lambda t.\ t)$
$\texttt{GSEQ}\ (\lambda t.\ p)$
$\texttt{GSEQ}\ e \Rightarrow \texttt{GSEQ}\ (\lambda t.\ a..e\ t)$
$\texttt{GSEQ}\ e_1 \wedge \texttt{GSEQ}\ e_2 \Rightarrow \texttt{GSEQ}\ (\lambda t.\ a_1..e_1\ t + a_2..e_2\ t)$
[GSEQ_rules]

Contraction: formal definition

$$
\begin{array}{l}
\vdash \texttt{CONTRACTION}\ \mathit{Con} \iff \\
\quad \forall E\ E'.\\
\qquad \mathit{Con}\ E\ E' \Rightarrow\\
\qquad (\forall l.\\
\qquad\quad (\forall E_1.\\
\qquad\qquad E\ \texttt{-label}\ l\texttt{->}\ E_1 \Rightarrow\\
\qquad\qquad \exists E_2.\ E'\ \texttt{-label}\ l\texttt{->}\ E_2 \land \mathit{Con}\ E_1\ E_2) \land\\
\qquad\quad \forall E_2.\\
\qquad\qquad E'\ \texttt{-label}\ l\texttt{->}\ E_2 \Rightarrow \exists E_1.\ E\ \texttt{=label}\ l\texttt{=>}\ E_1 \land E_1 \approx E_2) \land\\
\qquad (\forall E_1.\\
\qquad\quad E\ \texttt{-}\tau\texttt{->}\ E_1 \Rightarrow \mathit{Con}\ E_1\ E' \lor \exists E_2.\ E'\ \texttt{-}\tau\texttt{->}\ E_2 \land \mathit{Con}\ E_1\ E_2) \land\\
\qquad \forall E_2.\ E'\ \texttt{-}\tau\texttt{->}\ E_2 \Rightarrow \exists E_1.\ E \stackrel{\epsilon}{\Rightarrow} E_1 \land E_1 \approx E_2 \qquad [\texttt{CONTRACTION}]
\end{array}
$$

$\vdash P \succeq_{bis} Q \iff \exists \mathit{Con}.\ \mathit{Con}\ P\ Q \land \texttt{CONTRACTION}\ \mathit{Con}$ [contracts_thm]

$\vdash P \succeq_{bis} Q \Rightarrow P \approx Q$ [contracts_IMP_WEAK_EQUIV]

$\succeq_{\mathrm{bis}}$ is preorder and precongruence:

⊢ PreOrder (contracts) [contracts_PreOrder]

⊢ precongruence1 (contracts) [contracts_precongruence]

Rooted contraction: formal definition

$$
\begin{array}{l}
\vdash E \succeq^{c}_{bis} E' \iff \\
\quad \forall u. \\
\qquad (\forall E_1.\ E \ -u\rightarrow\ E_1 \Rightarrow \exists E_2.\ E' \ -u\rightarrow\ E_2 \wedge E_1 \succeq_{bis} E_2) \wedge \\
\qquad \forall E_2.\ E' \ -u\rightarrow\ E_2 \Rightarrow \exists E_1.\ E \ =u\Rightarrow\ E_1 \wedge E_1 \approx E_2
\end{array}
$$

[OBS_contracts]

$\vdash E \succeq^{c}_{bis} E' \Rightarrow E \succeq_{bis} E'$ [OBS_contracts_IMP_contracts]

$\vdash$ `PreOrder OBS_contracts` [OBS_contracts_PreOrder]

$\vdash$ `precongruence OBS_contracts` [OBS_contracts_precongruence]

Inspired by the definition rooted bisimilarity (not recursive, built upon non-rooted relation), with candidates quickly checked by theorem prover on its transitivity)

$$
\begin{array}{l}
\vdash E \approx^{c} E' \iff \\
\quad \forall u. \\
\qquad (\forall E_1.\ E \ -u\rightarrow\ E_1 \Rightarrow \exists E_2.\ E' \ =u\Rightarrow\ E_2 \wedge E_1 \approx E_2) \wedge \\
\qquad \forall E_2.\ E' \ -u\rightarrow\ E_2 \Rightarrow \exists E_1.\ E \ =u\Rightarrow\ E_1 \wedge E_1 \approx E_2
\end{array}
$$

All five unique solution theorems

1. STRONG_UNIQUE_SOLUTION:

   $\vdash \texttt{WG}\ E \Rightarrow \forall P\ Q.\ P \sim E\ P \wedge Q \sim E\ Q \Rightarrow P \sim Q$

2. WEAK_UNIQUE_SOLUTION:

   $\vdash \texttt{SG}\ E \wedge \texttt{GSEQ}\ E \Rightarrow \forall P\ Q.\ P \approx E\ P \wedge Q \approx E\ Q \Rightarrow P \approx Q$

3. OBS_UNIQUE_SOLUTION:

   $\vdash \texttt{SG}\ E \wedge \texttt{SEQ}\ E \Rightarrow \forall P\ Q.\ P \approx^c E\ P \wedge Q \approx^c E\ Q \Rightarrow P \approx^c Q$

4. UNIQUE_SOLUTION_OF_CONTRACTIONS:

   $\vdash \texttt{WGS}\ E \Rightarrow \forall P\ Q.\ P \succeq_{bis} E\ P \wedge Q \succeq_{bis} E\ Q \Rightarrow P \approx Q$

5. UNIQUE_SOLUTION_OF_ROOTED_CONTRACTIONS:

   $\vdash \texttt{WG}\ E \Rightarrow \forall P\ Q.\ P \succeq^c_{bis} E\ P \wedge Q \succeq^c_{bis} E\ Q \Rightarrow P \approx^c Q$

Coarsest congruence contained in $\approx$

Theorem

Assumping p and q do not use all labels, i.e. $\mathrm{fn}(p) \cup \mathrm{fn}(q) \neq \mathscr{L}$,

$$p \approx^{\mathrm{c}} q \iff (\forall r.\ p + r \approx q + r)\ .$$

Our formalised version (with slightly weaker assumptions):

$\vdash$ `free_action` $p \wedge$ `free_action` $q \Rightarrow (p \approx^{c} q \iff \forall r.\ p + r \approx q + r)$ [COARSEST_CONGR_THM]

where

`free_action` $p \iff \exists a.\ \forall p'.\ \neg(p$ `=label` $a \Rightarrow p')$ [free_action_def]

Coarsest congruence contained in $\approx$ (van Glabbeek's method)

Lemma

Given processes p and q, if there's a special process $k(p, q)$, then the hard part ($\Leftarrow$) of "coarsest congruence" theorem holds without classic assumption.

$\vdash \forall p\ q.\ (\exists k.\ \texttt{STABLE}\ k \land (\forall p'\ u.\ p = u \Rightarrow p' \Rightarrow \neg(p' \approx k)) \land \forall q'\ u.\ q = u \Rightarrow q' \Rightarrow \neg(q' \approx k)) \Rightarrow (\forall r.\ p + r \approx q + r) \Rightarrow p \approx^c q$ [PROP3_COMMON]

$\texttt{STABLE}\ E \iff \forall u\ E'.\ E - u \rightarrow E' \Rightarrow u \neq \tau$ [STABLE]

Definition (Arbitrary non-bisimir processes (Klop) - finite version)

KLOP_def:

$\vdash (\forall a.\ \texttt{KLOP}\ a\ 0 = \texttt{nil}) \land \forall a\ n.\ \texttt{KLOP}\ a\ (\texttt{SUC}\ n) = \texttt{KLOP}\ a\ n + \texttt{label}\ a..\texttt{KLOP}\ a\ n$

KLOP_PROP2':

$\vdash m < n \Rightarrow \neg(\texttt{KLOP}\ a\ m \approx \texttt{KLOP}\ a\ n)$

Lemma

$\vdash \texttt{finite\_state}\ p \land \texttt{finite\_state}\ q \Rightarrow \exists k.\ \texttt{STABLE}\ k \land (\forall p'\ u.\ p = u \Rightarrow p' \Rightarrow \neg(p' \approx k)) \land \forall q'\ u.\ q = u \Rightarrow q' \Rightarrow \neg(q' \approx k)$ [KLOP_LEMMA_FINITE]

Coarsest precongruence contained in $\succeq^{c}_{\mathrm{bis}}$

Theorem

$\vdash$ `free_action` $p \land$ `free_action` $q \Rightarrow (p \succeq^{c}_{bis} q \iff \forall r.\ p + r \succeq_{bis} q + r)$ [COARSEST_PRECONGR_THM']

Thus the current definition of $\succeq^{c}_{\mathrm{bis}}$ is the best possible one.

Another version following van Glabbeek's proof:

$\vdash$ `finite_state` $p \land$ `finite_state` $q \Rightarrow (p \succeq^{c}_{bis} q \iff \forall r.\ p + r \succeq_{bis} q + r)$ [COARSEST_PRECONGR_FINITE]

`finite_state` $p \iff$ `FINITE (NODES` p`)` [finite_state_def]

`NODES` p = $\{q \mid$ `Reach` $p\ q\}$ [NODES_def]

`Reach` = $(\lambda E\ E'.\ \exists u.\ E \overset{u}{\longrightarrow} E')^*$ [Reach_def]

Toward multi-variable equations

The multi-variable case is a “routine” adaptation in informal proofs; theorems on multi-variable equation cannot be proved by their single-variable version. Current idea: reuse the guardedness definition of single-variable equation (i.e. multi-hole contexts)

Examples

$$
\begin{aligned}
E[X; Y] &= a.\, X + b.\, X + c.\, Y + d.\, Y \\
E_1[\cdot] &= a.\, [] + b.\, X + c.\, Y + d.\, Y \\
E_2[\cdot] &= a.\, X + b.\, [] + c.\, Y + d.\, Y \\
E_3[\cdot] &= a.\, [] + b.\, [] + c.\, Y + d.\, Y \\
E[X; Y] &= E_1[X] = E_2[X] = E_3[X]
\end{aligned}
$$

Definition

`weakly_guarded1` $E \iff \forall X.\ X \in$ `FV` $E \Rightarrow \forall e.$ `CONTEXT` $e \wedge (e$ `(var` X`)` = $E) \Rightarrow$ `WG` e [weakly_guaded1_def]

Future directions

1. Formalizing unique-solution theorems under multi-variable equations/contractions (represented as CCS terms with free variables).
2. Formalizing CCS theorems related to free/bound names and free/bound variables. (learning from experiences from π-calculi formalisations)
3. Formalizing Sangiorgi's 2017 work (Divergence and Unique Solution of Equations) and other CCS results in frontier.
4. Decision procedure for various bisimilarity. (Concurrency Workbench in HOL4)
5. CCS with arbitrary sums; Deeper look at the "coarsest (pre)congruence" theorem without classic assumptions.
6. Connecting CCS Theory with Graph Theory (LTS as digraph) and Probability Theory.

Conclusions

1. Now we have a *more* complete formalisation of CCS with an archive of rigous formal proofs of related theorems and lemmas in textbook.
2. Sangiorgi's theorem on unique solution of contractions is formally verified and slightly extended. (frontier)
3. *Sometimes* formalising a theory helps in finding new interesting results or refining previously known results.
4. This work could be a template or working basis for CCS extensions or other process calculi, even in other theorem provers than HOL.

This CCS formalisation will be continuously maintained as part of HOL4 official examples to make sure its long-term availability.