

# Formalized bisimulation-up-to

Chun Tian

Scuola di Scienze, Università di Bologna

chun.tian@studio.unibo.it

Numero di matricola: 0000735539

**Abstract.** Basic results for “bisimulation up to  $\sim$  and  $\approx$ ” are formalized in this work. The work is a necessary path towards Milner’s “unique solutions of equations” theorems.

## 1 Introduction

This is another preliminary thesis work, after [1], in which basic results for “bisimulation up to  $\sim$ ” were already proved. Both this short paper and [1] are based on the author’s previous two projects [2] [3], which are finally based on the work of Monica Nesi [4] during 1992-1995.

Here we first added a few more common results for the strong bisimulation cases, then turned to the (weak) bisimulation up to  $\approx$  and have proved all needed results towards Milner’s “unique solutions of equations” theorems.

It turned out that, the proof size for the weak bisimulation case, although essentially the same as in the strong bisimulation case, is almost 5 times longer, and we have to prove a large amount of lemmas to support the “short” proof of the final lemma and theorem.

## 2 Bisimulation up to $\sim$

In previous paper [1], we have shown our definition of “strong bisimulation upto  $\sim$ ”:

**Definition 1.** (*Strong bisimulation up to  $\sim$* )

$$\begin{aligned} \vdash \text{STRONG\_BISIM\_UPTO } Bsm &\iff \\ \forall E \ E'. & \\ Bsm \ E \ E' \Rightarrow & \\ \forall u. & \\ (\forall E_1. & \\ E \ --u-> E_1 \Rightarrow & \\ \exists E_2. & \\ E' \ --u-> E_2 \wedge & \\ (\text{STRONG\_EQUIV} \circ_r Bsm \circ_r \text{STRONG\_EQUIV}) \ E_1 \ E_2) \wedge & \\ \forall E_2. & \\ E' \ --u-> E_2 \Rightarrow & \\ \exists E_1. & \\ E \ --u-> E_1 \wedge & \\ (\text{STRONG\_EQUIV} \circ_r Bsm \circ_r \text{STRONG\_EQUIV}) \ E_1 \ E_2 & \end{aligned}$$

New additions for this part, are the following common properties:

**Proposition 1.** *Properties of “strong bisimulation upto  $\sim$ ”*

1. *Identity relation is a “strong bisimulation upto  $\sim$ ”:*

$$\vdash \text{STRONG\_BISIM\_UPTO } (\lambda x \ y. \ x = y)$$

2. *The converse of a “strong bisimulation upto  $\sim$ ” is still “strong bisimulation upto  $\sim$ ”:*

$$\vdash \text{STRONG\_BISIM\_UPTO } Wbsm \Rightarrow \text{STRONG\_BISIM\_UPTO } (\lambda x \ y. \ Wbsm \ y \ x)$$

And we have proved the following lemma, which establishes the relationship between “strong bisimulation upto  $\sim$ ” and “strong bisimulation”:

**Lemma 1.** *If  $\mathcal{S}$  is a strong bisimulation up to  $\sim$ , then  $\sim \mathcal{S} \sim$  is a strong bisimulation:*

$$\vdash \text{STRONG\_BISIM\_UPTO } Bsm \Rightarrow \\ \text{STRONG\_BISIM } (\text{STRONG\_EQUIV } \circ_r Bsm \circ_r \text{STRONG\_EQUIV})$$

*Proof.* The idea is to fix two process  $E$  and  $E'$ , which satisfies  $E \sim \circ Bsm \circ E'$ , then check it for the definition of strong bisimulation: for all  $E_1$  such that  $E \xrightarrow{-u-} E_1$ , there exists  $E_2''$  such that  $E' \xrightarrow{-u-} E_2''$  (the other side is totally symmetric), as shown in the following graph:

$$\begin{array}{ccccccc} E & \xrightarrow{\sim} & \exists y' & \xrightarrow{Bsm} & \exists y & \xrightarrow{\sim} & E' \\ \downarrow \forall u & & \swarrow u & & \searrow u & & \downarrow u \\ \forall E_1 & \xrightarrow{\sim} & \exists E_2 & \xrightarrow{\sim} & \exists y''' & \xrightarrow{Bsm} & \exists y'' & \xrightarrow{\sim} & \exists E_2' & \xrightarrow{\sim} & \exists E_2'' \end{array}$$

During the proof, needed lemmas are the definition of “bisimulation up to  $\sim$ ” (for expanding “ $y' Bsm y$ ” into “ $E_2 \sim y''' Bsm y'' \sim E_2'$ ”), plus the property (\*) and transitivity of strong equivalence.

Based on above lemma, we then easily proved the following proposition:

**Proposition 2.** *If  $\mathcal{S}$  is a strong bisimulation up to  $\sim$ , then  $\mathcal{S} \subseteq \sim$ :*

$$\vdash \text{STRONG\_BISIM\_UPTO } Bsm \Rightarrow Bsm \subseteq_r \text{STRONG\_EQUIV}$$

Hence, to prove  $P \sim Q$ , we only have to find a strong bisimulation up to  $\sim$  which contains  $(P, Q)$ .

### 3 Bisimulation up to $\approx$

The concept of bisimulation up to  $\approx$  is defined in the following way:

**Definition 2.** *(Bisimulation up to  $\approx$ )*

$$\begin{aligned} \vdash \text{WEAK\_BISIM\_UPTO } Wbsm &\iff \\ \forall E \ E'. & \\ Wbsm \ E \ E' &\Rightarrow \\ (\forall l. & \\ (\forall E_1. & \\ E \text{ --label } l \rightarrow E_1 &\Rightarrow \\ \exists E_2. & \\ E' \text{ ==label } l \Rightarrow E_2 \wedge & \\ (\text{WEAK\_EQUIV } \circ_r Wbsm \circ_r \text{STRONG\_EQUIV}) \ E_1 \ E_2) \wedge & \\ \forall E_2. & \\ E' \text{ --label } l \rightarrow E_2 &\Rightarrow \\ \exists E_1. & \\ E \text{ ==label } l \Rightarrow E_1 \wedge & \\ (\text{STRONG\_EQUIV } \circ_r Wbsm \circ_r \text{WEAK\_EQUIV}) \ E_1 \ E_2) \wedge & \\ (\forall E_1. & \\ E \text{ --}\tau\text{--} E_1 &\Rightarrow \\ \exists E_2. & \\ E' \xrightarrow{\epsilon} E_2 \wedge (\text{WEAK\_EQUIV } \circ_r Wbsm \circ_r \text{STRONG\_EQUIV}) \ E_1 \ E_2) \wedge & \\ \forall E_2. & \\ E' \text{ --}\tau\text{--} E_2 &\Rightarrow \\ \exists E_1. \ E \xrightarrow{\epsilon} E_1 \wedge (\text{STRONG\_EQUIV } \circ_r Wbsm \circ_r \text{WEAK\_EQUIV}) \ E_1 \ E_2 & \end{aligned}$$

However, there're a few things to be noticed:

1. In HOL4, the big “O” notion as relation composition has different orders with usual Math notation: *the right-most relation takes the input argument first*, which is actually the case for function composition:  $(f \circ g)(x) = f(g(x))$ . Thus in all HOL-generated terms like

$$(\text{WEAK\_EQUIV} \circ_r \text{Wbsm} \circ_r \text{STRONG\_EQUIV}) \ E_1 \ E_2$$

in this paper, it should be understood like “ $E_1 \sim y \text{Wbsm } y' \approx E_2$ ”. (There was no such issues for the strong bisimulation cases, because we had  $\sim$  on both side)

2. The original definition in Milner’s book [5], in which he used  $\approx \circ R \circ \approx$  in all places in above definition, is wrong. The reason has been explained in Gorrieri’s book [6] (page 65), that the resulting relation may not be a subset of  $\approx$ ! Thus we have used the definition from Gorrieri’s book, with the definition in Sangiorgi’s book [7] (page 115) doubly confirmed.
3. Some authors (e.g. Prof. Sangiorgi) uses the notions like  $P \stackrel{\mu}{\Rightarrow} P'$  to represent special case that,  $P \stackrel{\tau}{\Rightarrow} P'$  when  $\mu = \tau$  (i.e. it’s possible that  $P = Q$ ). Such notions are concise, but inconvenient to use in formalization work, because the EPS transition, in many cases, has very different characteristics. Thus we have above long definition but easier to use when proving all needed results.

Two basic properties to help understanding “bisimulation up to  $\approx$ ”:

**Lemma 2.** (*Properties of bisimulation up to  $\approx$* )

1. The identity relation is “bisimulation up to  $\approx$ ”:

$$\vdash \text{WEAK\_BISIM\_UPTO} \ (\lambda x \ y. \ x = y)$$

2. The converse of a “bisimulation up to  $\approx$ ” is still “bisimulation up to  $\approx$ ”:

$$\vdash \text{WEAK\_BISIM\_UPTO} \ \text{Wbsm} \Rightarrow \text{WEAK\_BISIM\_UPTO} \ (\lambda x \ y. \ \text{Wbsm} \ y \ x)$$

Now we want to prove the following main lemma:

**Lemma 3.** *If  $\mathcal{S}$  is a bisimulation up to  $\approx$ , then  $\approx \mathcal{S} \approx$  is a bisimulation.*

$$\vdash \text{WEAK\_BISIM\_UPTO} \ \text{Wbsm} \Rightarrow \\ \text{WEAK\_BISIM} \ (\text{WEAK\_EQUIV} \circ_r \text{Wbsm} \circ_r \text{WEAK\_EQUIV})$$

*Proof.* Milner’s books simply said that the proof is “analogous” to the same lemma for strong bisimulation. This is basically true, from left to right (for visible transitions):

$$\begin{array}{ccccccc} E & \xrightarrow{\approx} & \exists y' & \xrightarrow{\text{Wbsm}} & \exists y & \xrightarrow{\approx} & E' \\ \downarrow \forall l & & \swarrow l & & \searrow l & & \downarrow l \\ \forall E_1 & \xrightarrow{\approx} & \exists E_2 & \xrightarrow{\approx} & \exists y''' & \xrightarrow{\text{Wbsm}} & \exists y'' & \xrightarrow{\approx} & \exists E'_2 & \xrightarrow{\approx} & \exists E''_2 \end{array}$$

There’s a little difficulty, however. Given  $y \approx E'$  and  $y \stackrel{l}{\Rightarrow} E'_2$ , the existence of  $E''_2$  doesn’t follow directly from the definition or property (\*) of weak equivalence. Instead, we have to prove a lemma (to be presented below) to finish this last step.

More difficulties appear from right to left:

$$\begin{array}{ccccccc} E & \xrightarrow{\approx} & \exists y' & \xrightarrow{\text{Wbsm}} & \exists y & \xrightarrow{\approx} & E' \\ \downarrow l & & \swarrow l & & \searrow l & & \downarrow \forall l \\ \exists E''_1 & \xrightarrow{\approx} & \exists E'_1 & \xrightarrow{\approx} & \exists y''' & \xrightarrow{\text{Wbsm}} & \exists y'' & \xrightarrow{\approx} & \exists E_1 & \xrightarrow{\approx} & \forall E_2 \end{array}$$

The problem is, given  $y' \text{Bsm } y$  and  $y \stackrel{l}{\Rightarrow} E_1$ , the existence of  $E'_1$  doesn’t follow directly from the definition of “bisimulation up to  $\approx$ ”, instead this result must be proved (to be presented below) and the proof is non-trivial.

The other two cases concerning  $\tau$ -transitions:

$$\begin{array}{ccccccc}
E & \xrightarrow{\approx} & \exists y' & \xrightarrow{Wbsm} & \exists y & \xrightarrow{\approx} & E' \\
\downarrow \tau & & \searrow \epsilon & & \searrow \epsilon & & \downarrow \epsilon \\
\forall E_1 & \xrightarrow{\approx} & \exists E_2 & \xrightarrow{\sim} & \exists y''' & \xrightarrow{Wbsm} & \exists y'' & \xrightarrow{\approx} & \exists E_2' & \xrightarrow{\approx} & \exists E_2''
\end{array}$$

in which the EPS transition bypass of weak equivalence (from  $E_2'$  to  $E_2''$ ) must be proved, and

$$\begin{array}{ccccccc}
E & \xrightarrow{\approx} & \exists y' & \xrightarrow{Wbsm} & \exists y & \xrightarrow{\approx} & E' \\
\downarrow \epsilon & & \searrow \epsilon & & \searrow \epsilon & & \downarrow \tau \\
\exists E_1'' & \xrightarrow{\approx} & \exists E_1' & \xrightarrow{\sim} & \exists y''' & \xrightarrow{Wbsm} & \exists y'' & \xrightarrow{\sim} & \exists E_1 & \xrightarrow{\approx} & \forall E_2
\end{array}$$

in which the EPS transition bypass for “bisimulation up to  $\approx$ ” (from  $E_1$  to  $E_1'$ ) must be proved as a lemma.

As a summary of all “difficulties”, here is a list of lemmas we have used to prove the previous lemma. Each lemma has also their “companion lemma” concerning the other directions (here we omit them):

**Lemma 4.** (Useful lemmas concerning the first weak transitions from  $\sim$ ,  $\approx$  and “bisimulation up to  $\approx$ ”)

**STRONG\_EQUIV\_EPS:**

$$\vdash E \sim E' \Rightarrow \forall E_1. E \xRightarrow{\epsilon} E_1 \Rightarrow \exists E_2. E' \xRightarrow{\epsilon} E_2 \wedge E_1 \sim E_2$$

**WEAK\_EQUIV\_EPS:**

$$\vdash E \approx E' \Rightarrow \forall E_1. E \xRightarrow{\epsilon} E_1 \Rightarrow \exists E_2. E' \xRightarrow{\epsilon} E_2 \wedge E_1 \approx E_2$$

**WEAK\_EQUIV\_WEAK\_TRANS\_label:**

$$\vdash E \approx E' \Rightarrow$$

$$\forall l E_1. E ==_{\text{label } l} E_1 \Rightarrow \exists E_2. E' ==_{\text{label } l} E_2 \wedge E_1 \approx E_2$$

**WEAK\_EQUIV\_WEAK\_TRANS\_tau:**

$$\vdash E \approx E' \Rightarrow \forall E_1. E ==_{\tau} E_1 \Rightarrow \exists E_2. E' \xRightarrow{\epsilon} E_2 \wedge E_1 \approx E_2$$

**WEAK\_BISIM\_UPTO\_EPS:**

$$\vdash \text{WEAK\_BISIM\_UPTO } Wbsm \Rightarrow$$

$$\forall E E'.$$

$$Wbsm E E' \Rightarrow$$

$$\forall E_1.$$

$$E \xRightarrow{\epsilon} E_1 \Rightarrow$$

$$\exists E_2. E' \xRightarrow{\epsilon} E_2 \wedge (\text{WEAK\_EQUIV } \circ_r Wbsm \circ_r \text{STRONG\_EQUIV}) E_1 E_2$$

**WEAK\_BISIM\_UPTO\_WEAK\_TRANS\_label:**

$$\vdash \text{WEAK\_BISIM\_UPTO } Wbsm \Rightarrow$$

$$\forall E E'.$$

$$Wbsm E E' \Rightarrow$$

$$\forall l E_1.$$

$$E ==_{\text{label } l} E_1 \Rightarrow$$

$$\exists E_2.$$

$$E' ==_{\text{label } l} E_2 \wedge$$

$$(\text{WEAK\_EQUIV } \circ_r Wbsm \circ_r \text{STRONG\_EQUIV}) E_1 E_2$$

*Proof.* (Proof sketch of above lemmas) The proof of **STRONG\_EQUIV\_EPS** and **WEAK\_EQUIV\_EPS** depends on the following “right induction theorem”<sup>1</sup> of the EPS transition:

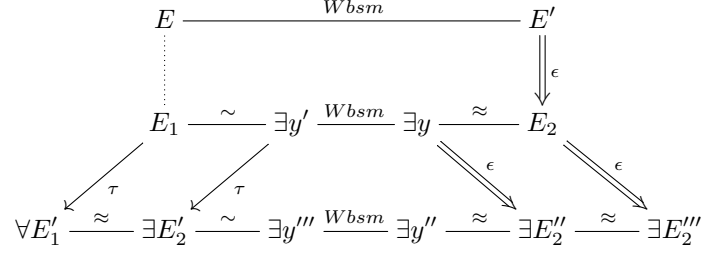
<sup>1</sup> Such induction theorems are part of HOL’s theorem library for all RTCs (reflexitive transitive closure). The proof of transitivity of observation congruence also heavily depends on this induction theorem, but in the work of Monica Nesi where the EPS relation is manually defined inductively, such an induction theorem is not available (and it’s not easy to prove it), as a result Monica Nesi couldn’t finish the proof for transitivity of observation congruence, which is incredible hard to prove without proving lemmas like **WEAK\_EQUIV\_EPS** first.

**EPS\_ind\_right:**

$$\vdash (\forall x. P \ x \ x) \wedge (\forall x \ y \ z. P \ x \ y \wedge y \ \text{--}\tau\text{--}\> z \Rightarrow P \ x \ z) \Rightarrow \forall x \ y. x \xrightarrow{\epsilon} y \Rightarrow P \ x \ y$$

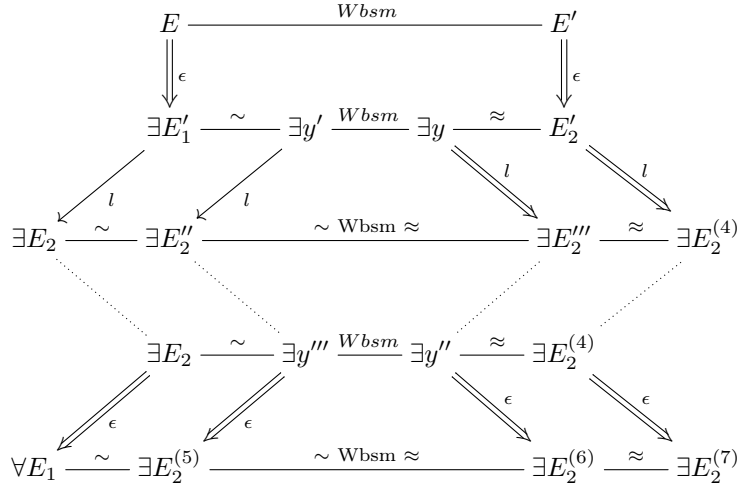
Basically, we need to prove that, if the lemma already holds for  $n - 1$   $\tau$ -transitions, it also holds for  $n$   $\tau$ -transitions.

The proof of **WEAK\_BISIM\_UPTO\_EPS** is also based on above induction theorem. The induction case of this proof can be sketched using the following graph:



The goal is to find  $E_2'''$  which satisfy  $E' \xrightarrow{\epsilon} E_2'''$ . As we can see from the graph, using the induction, now given  $y' \text{ Wbsm } y$  and  $y' \xrightarrow{\tau} E_2'$ , we can easily crossover the “bisimulation up to  $\approx$ ” and assert the existence of  $E_2''$  to finish the proof.

The proof of **WEAK\_BISIM\_UPTO\_WEAK\_TRANS\_label** is based on **WEAK\_BISIM\_UPTO\_EPS**. It is much more difficult, because an even bigger graph must be step-by-step constructed:



That is, for all  $E_1$  such that  $E \xrightarrow{l} E_1$  (which by definition of weak transitions exists  $E_1'$  and  $E_2$  such that  $E \xrightarrow{\epsilon} E_1'$ ,  $E_1' \xrightarrow{l} E_2$  and  $E_2 \xrightarrow{\epsilon} E_1$ ), we would like to finally find an  $E_2^{(7)}$  such that  $E' \xrightarrow{l} E_2^{(7)}$ . This process is long and painful, and we have to use **WEAK\_BISIM\_UPTO\_EPS** twice. The formal proof tries to build above graph by asserting the existences of each process step-by-step, until it finally reached to  $E_2^{(7)}$ . This proof is so-far the largest formal proof (in single branch) that the author ever met, before closing it has 26 assumptions which represents above graph:

?E2. E' ==label l=>> E2 /\ (WEAK\_EQUIV 0 Wbsm 0 STRONG\_EQUIV) E1 E2

- 
0. WEAK\_BISIM\_UPTO Wbsm
  1. Wbsm E E'
  2. E ==label l=>> E1
  3. EPS E E1'
  4. E1' --label l-> E2
  5. EPS E2 E1
  6. EPS E' E2'
  7. STRONG\_EQUIV E1' y'

```

8.  Wbsm y' y
9.  WEAK_EQUIV y E2'
10. y' --label l-> E2''
11. STRONG_EQUIV E2 E2''
12. y ==label l=>> E2'''
13. (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E2'' E2'''
14. E2' ==label l=>> E2''''
15. WEAK_EQUIV E2''' E2''''
16. STRONG_EQUIV E2 y'''
17. Wbsm y''' y''
18. WEAK_EQUIV y'' E2''''
19. EPS y''' E2''''
20. STRONG_EQUIV E1 E2''''
21. EPS y'' E2''''
22. (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E2'''' E2''''
23. EPS E2'''' E2''''
24. WEAK_EQUIV E2'''' E2''''
25. (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2''''

```

All above lemmas concern the cases from left and right (for all  $P$ , exists  $Q$  such that ...) To prove the other side (for all  $Q$  there exist  $P$  such that ...), there's no need to go over the painful proving process again, instead we can easily derive the other side by using `CONVERSE_WEAK_BISIM_UPTO`. For example, once above lemma `WEAK_BISIM_UPTO_WEAK_TRANS_label` is proved, it's trivial to get the following companion lemma:

**Lemma 5.** `WEAK_BISIM_UPTO_WEAK_TRANS_label'`:

$$\begin{aligned}
& \vdash \text{WEAK\_BISIM\_UPTO } Wbsm \Rightarrow \\
& \quad \forall E \ E'. \\
& \quad \quad Wbsm \ E \ E' \Rightarrow \\
& \quad \quad \forall l \ E_2. \\
& \quad \quad \quad E' ==label \ l \Rightarrow E_2 \Rightarrow \\
& \quad \quad \quad \exists E_1. \\
& \quad \quad \quad E ==label \ l \Rightarrow E_1 \wedge \\
& \quad \quad \quad (\text{STRONG\_EQUIV } \circ_r \ Wbsm \ \circ_r \ \text{WEAK\_EQUIV}) \ E_1 \ E_2
\end{aligned}$$

Finally, once the main lemma `WEAK_BISIM_UPTO_LEMMA`, the following final result can be easily proved, following the same idea in the proof of strong bisimulation cases:

**Theorem 1.** `WEAK_BISIM_UPTO_THM`:

$$\vdash \text{WEAK\_BISIM\_UPTO } Wbsm \Rightarrow Wbsm \subseteq_r \text{WEAK\_EQUIV}$$

## 4 Conclusions

So far we have proved all needed results for “bisimulation up to  $\approx$ ”, now we’re ready to proceed Milner’s “unique solutions of equations” theorem for weak bisimulation cases. But the work mentioned in this paper is also the basis to prove the further “unique solutions” theorem in Prof. Sangiorgi’s paper [8].

To the best of our knowledge, beside the work of Monica Nesi in Hol88 and the author’s porting work to HOL4 with extensions upto this paper, no other formalization ever reached the “bisimulation up to” concepts. Beside our work, currently the most comprehensive (publicly available) formalization of CCS is the formalization found in Isabelle/AFP<sup>2</sup> based on so-called “nominal logic”, done by Jesper Bengtson as part of his thesis (from the length (498 pages) of that thesis paper<sup>3</sup>, it must be a ph.D thesis.), but “bisimulation up to” are not part of the formalization. That student tried to cover not only CCS but also  $\pi$ -calculus and the so-called  $\psi$ -calculi (don’t know what it is), it seems that the goal is too big and as a result, none is really touching any deep

<sup>2</sup> <https://www.isa-afp.org/entries/CCS.html>

<sup>3</sup> <http://www.itu.dk/people/jebe/files/thesis.pdf>

result in these formal systems. The author, instead, would like to focus on CCS (at least before his graduation) and tries to touch the frontier through multiple projects on CCS formalization, with each one based on previous parts.

## References

1. Tian, C.: Unique solutions of equations for strong equivalence. (August 2017) 1–18
2. Tian, C.: A Formalization of the Process Algebra CCS in HOL4. arXiv.org, <http://arxiv.org/abs/1705.07313v2> (May 2017)
3. Tian, C.: Further Formalization of the Process Algebra CCS in HOL4. arXiv.org, <http://arxiv.org/abs/1707.04894v2> (July 2017)
4. Nesi, M.: A formalization of the process algebra CCS in high order logic. Technical report, University of Cambridge, Computer Laboratory (1992)
5. Milner, R.: Communication and concurrency. Prentice Hall (1989)
6. Gorrieri, R., Versari, C.: Introduction to Concurrency Theory. Transition Systems and CCS. Springer, Cham (September 2015)
7. Sangiorgi, D.: Introduction to Bisimulation and Coinduction. Cambridge University Press (October 2011)
8. Sangiorgi, D.: Equations, contractions, and unique solutions. ACM SIGPLAN Notices (2015)

## Appendix: proof scripts

A digital version of original proof scripts can be viewed and downloaded from the following GitHub address:

<https://github.com/binghe/informatica-public/tree/master/pre-thesis>

```

1  (*)
2  * Copyright 2016-2017 University of Bologna (Author: Chun Tian)
3  *)
4
5  open HolKernel Parse boolLib bossLib;
6
7  open pred_setTheory relationTheory pairTheory sumTheory listTheory;
8  open prim_recTheory arithmeticTheory combinTheory;
9
10 open CCSLib CCSTheory CCSSyntax CCSConv;
11 open StrongEQTheory StrongEQLib StrongLawsTheory StrongLawsConv;
12 open WeakEQTheory WeakEQLib WeakLawsTheory WeakLawsConv;
13 open ObsCongrTheory ObsCongrLib ObsCongrLawsTheory ObsCongrConv;
14
15 val _ = new_theory "BisimulationUpto";
16
17 (*****)
18 (*)
19 (*)      Strong Bisimulation Upto ~
20 (*)
21 (*****)
22
23 (* Define the strong bisimulation relation up to STRONG_EQUIV *)
24 val STRONG_BISIM_UPTO = new_definition (
25   "STRONG_BISIM_UPTO",
26   "STRONG_BISIM_UPTO (Bsm :('a, 'b) simulation) =
27     !E E'.
28       Bsm E E' ==>
```

```

29      !u. (!E1. TRANS E u E1 ==>
30          ?E2. TRANS E' u E2 /\ (STRONG_EQUIV 0 Bsm 0 STRONG_EQUIV) E1 E2) /\
31          (!E2. TRANS E' u E2 ==>
32              ?E1. TRANS E u E1 /\ (STRONG_EQUIV 0 Bsm 0 STRONG_EQUIV) E1 E2)');
33
34 val IDENTITY_STRONG_BISIM_UPTO_lemma = store_thm (
35     "IDENTITY_STRONG_BISIM_UPTO_lemma",
36     '(!E. (STRONG_EQUIV 0 (\x y. x = y) 0 STRONG_EQUIV) E E',
37     GEN_TAC >> REWRITE_TAC [O_DEF] >> BETA_TAC
38     >> NTAC 2 (Q.EXISTS_TAC 'E' >> REWRITE_TAC [STRONG_EQUIV_REFL]));
39
40 val IDENTITY_STRONG_BISIM_UPTO = store_thm (
41     "IDENTITY_STRONG_BISIM_UPTO", '(!STRONG_BISIM_UPTO (\x y. x = y)',
42     PURE_ONCE_REWRITE_TAC [STRONG_BISIM_UPTO]
43     >> BETA_TAC
44     >> REPEAT STRIP_TAC (* 2 sub-goals *)
45     >| [ (* goal 1 *)
46         ASSUME_TAC (REWRITE_RULE [ASSUME 'E:(\'a, \'b) CCS = E',
47             (ASSUME 'TRANS E u E1')] \
48         EXISTS_TAC 'E1 :('a, \'b) CCS' \
49         ASM_REWRITE_TAC [] \
50         REWRITE_TAC [IDENTITY_STRONG_BISIM_UPTO_lemma],
51         (* goal 2 *)
52         PURE_ONCE_ASM_REWRITE_TAC [] \
53         EXISTS_TAC 'E2 :('a, \'b) CCS' \
54         ASM_REWRITE_TAC [] \
55         REWRITE_TAC [IDENTITY_STRONG_BISIM_UPTO_lemma] ]);
56
57 val CONVERSE_STRONG_BISIM_UPTO_lemma = Q.prove (
58     '!Wbsm E E'. (STRONG_EQUIV 0 (\x y. Wbsm y x) 0 STRONG_EQUIV) E E' =
59     (STRONG_EQUIV 0 Wbsm 0 STRONG_EQUIV) E' E',
60     rpt GEN_TAC
61     >> EQ_TAC (* 2 sub-goals here *)
62     >| [ (* goal 1 (of 2) *)
63         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \
64         'STRONG_EQUIV y' E' by PROVE_TAC [STRONG_EQUIV_SYM] \
65         'STRONG_EQUIV E' y' by PROVE_TAC [STRONG_EQUIV_SYM] \
66         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \
67         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [],
68         (* goal 2 (of 2) *)
69         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \
70         'STRONG_EQUIV E y' by PROVE_TAC [STRONG_EQUIV_SYM] \
71         'STRONG_EQUIV y' E' by PROVE_TAC [STRONG_EQUIV_SYM] \
72         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \
73         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] ]);
74
75 val CONVERSE_STRONG_BISIM_UPTO = store_thm (
76     "CONVERSE_STRONG_BISIM_UPTO",
77     '(!Wbsm. STRONG_BISIM_UPTO Wbsm ==> STRONG_BISIM_UPTO (\x y. Wbsm y x)',
78     GEN_TAC
79     >> PURE_ONCE_REWRITE_TAC [STRONG_BISIM_UPTO]
80     >> BETA_TAC
81     >> rpt STRIP_TAC
82     >> RES_TAC (* 2 sub-goals here *)
83     >| [ (* goal 1 (of 2) *)
84         Q.EXISTS_TAC 'E1' >> ASM_REWRITE_TAC [] \
85         REWRITE_TAC [CONVERSE_STRONG_BISIM_UPTO_lemma] \
86         ASM_REWRITE_TAC [],
87         (* goal 2 (of 2) *)
88         Q.EXISTS_TAC 'E2' >> ASM_REWRITE_TAC [] \
89         REWRITE_TAC [CONVERSE_STRONG_BISIM_UPTO_lemma] \

```



```

90     ASM_REWRITE_TAC [] ]);
91
92 val STRONG_BISIM_UPTO_LEMMA = store_thm (
93   "STRONG_BISIM_UPTO_LEMMA",
94   '!Bsm. STRONG_BISIM_UPTO Bsm ==> STRONG_BISIM (STRONG_EQUIV O Bsm O STRONG_EQUIV)',
95   GEN_TAC
96 >> REWRITE_TAC [STRONG_BISIM, O_DEF]
97 >> rpt STRIP_TAC (* 2 sub-goals here *)
98 >| [ (* goal 1 (of 2) *)
99     Q.PAT_X_ASSUM 'STRONG_EQUIV E y',
100     (STRIP_ASSUME_TAC o (ONCE_REWRITE_RULE [PROPERTY_STAR])) \\  

101     POP_ASSUM (STRIP_ASSUME_TAC o (Q.SPEC 'u')) \\  

102     POP_ASSUM K_TAC \\  

103     RES_TAC \\  

104     Q.PAT_X_ASSUM 'STRONG_BISIM_UPTO Bsm',
105     (STRIP_ASSUME_TAC o (REWRITE_RULE [STRONG_BISIM_UPTO])) \\  

106     RES_TAC \\  

107     NTAC 4 (POP_ASSUM K_TAC) \\  

108     POP_ASSUM (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\  

109     Q.PAT_X_ASSUM 'STRONG_EQUIV y E',
110     (STRIP_ASSUME_TAC o (ONCE_REWRITE_RULE [PROPERTY_STAR])) \\  

111     POP_ASSUM (STRIP_ASSUME_TAC o (Q.SPEC 'u')) \\  

112     POP_ASSUM K_TAC \\  

113     POP_ASSUM (STRIP_ASSUME_TAC o
114       (fn th => MATCH_MP th (ASSUME 'TRANS y u E2')) \\  

115   (***)
116       E      ~      y'      Bsm      y      ~      E'
117       /      /      \      /
118       u      u      u      u
119       /      /      \      /
120       E1 ~ E2 ~ y''' Bsm y'' ~ E2' ~ E2''
121   (***)
122   'STRONG_EQUIV E1 y''' by PROVE_TAC [STRONG_EQUIV_TRANS] \\  

123   'STRONG_EQUIV y'' E2''' by PROVE_TAC [STRONG_EQUIV_TRANS] \\  

124   Q.EXISTS_TAC 'E2''' >> ASM_REWRITE_TAC [] \\  

125   Q.EXISTS_TAC 'y''' >> ASM_REWRITE_TAC [] \\  

126   Q.EXISTS_TAC 'y''' >> ASM_REWRITE_TAC [],
127   (* goal 2 (of 2) *)
128   Q.PAT_X_ASSUM 'STRONG_EQUIV y E',
129   (STRIP_ASSUME_TAC o (ONCE_REWRITE_RULE [PROPERTY_STAR])) \\  

130   POP_ASSUM (STRIP_ASSUME_TAC o (Q.SPEC 'u')) \\  

131   Q.PAT_X_ASSUM '!E1. TRANS y u E1 ==> P' K_TAC \\  

132   RES_TAC \\  

133   Q.PAT_X_ASSUM 'STRONG_BISIM_UPTO Bsm',
134   (STRIP_ASSUME_TAC o (REWRITE_RULE [STRONG_BISIM_UPTO])) \\  

135   RES_TAC \\  

136   NTAC 2 (POP_ASSUM K_TAC) \\  

137   POP_ASSUM (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\  

138   Q.PAT_X_ASSUM 'STRONG_EQUIV E y',
139   (STRIP_ASSUME_TAC o (ONCE_REWRITE_RULE [PROPERTY_STAR])) \\  

140   POP_ASSUM (STRIP_ASSUME_TAC o (Q.SPEC 'u')) \\  

141   Q.PAT_X_ASSUM '!E1. TRANS E u E1 ==> P' K_TAC \\  

142   POP_ASSUM (STRIP_ASSUME_TAC o
143     (fn th => MATCH_MP th (ASSUME 'TRANS y' u E1')) \\  

144   (***)
145       E      ~      y'      Bsm      y      ~      E'
146       /      /      \      /
147       u      u      u      u
148       /      /      \      /
149       E1'' ~ E1' ~ y''' Bsm y'' ~ E1 ~ E2
150   (***)

```

```

151     'STRONG_EQUIV E1'' y'''' by PROVE_TAC [STRONG_EQUIV_TRANS] \\
152     'STRONG_EQUIV y'' E2' by PROVE_TAC [STRONG_EQUIV_TRANS] \\
153     Q.EXISTS_TAC 'E1'' >> ASM_REWRITE_TAC [] \\
154     Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] \\
155     Q.EXISTS_TAC 'y'''' >> ASM_REWRITE_TAC [] ]);
156
157 val STRONG_BISIM_UPTO_THM = store_thm (
158     "STRONG_BISIM_UPTO_THM",
159     '!Bsm. STRONG_BISIM_UPTO Bsm ==> Bsm RSUBSET STRONG_EQUIV',
160     rpt STRIP_TAC
161 >> IMP_RES_TAC STRONG_BISIM_UPTO_LEMMA
162 >> IMP_RES_TAC STRONG_BISIM_SUBSET_STRONG_EQUIV
163 >> Suff 'Bsm RSUBSET (STRONG_EQUIV 0 Bsm 0 STRONG_EQUIV)'
164 >- ( DISCH_TAC \\
165     Know 'transitive ((RSUBSET) :('a, 'b) simulation -> ('a, 'b) simulation -> bool)'
166     >- PROVE_TAC [RSUBSET_WeakOrder, WeakOrder] \\
167     RW_TAC std_ss [transitive_def] >> RES_TAC )
168 >> KILL_TAC
169 >> REWRITE_TAC [RSUBSET, 0_DEF]
170 >> rpt STRIP_TAC
171 >> 'STRONG_EQUIV x x /\ STRONG_EQUIV y y' by PROVE_TAC [STRONG_EQUIV_REFL]
172 >> Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC []
173 >> Q.EXISTS_TAC 'x' >> ASM_REWRITE_TAC []
174 (* Hence, to prove  $P \sim Q$ , we only have to find a strong bisimulation up to  $\sim$ 
175    which contains  $(P, Q)$  *));
176
177 (*****
178 (*)
179 (*)
180 (*)
181 *)
182
183 (* NOTE: the definition in Milner's book [1] is wrong, we use the one in Gorrieri's book [2]
184    double-confirmed with Sangiorgi's book [3].
185
186    IMPORTANT: in HOL's big "0", the second argument to composition acts on the "input" first
187    so we need to revert the order of  $(?a \ 0 \ Wbsm \ 0 \ ?b)$  when  $?a$  and  $?b$  are different.
188 *)
189 val WEAK_BISIM_UPTO = new_definition (
190     "WEAK_BISIM_UPTO",
191     '!E E'.
192     Wbsm E E' ==>
193     (!l.
194         (!E1. TRANS E (label l) E1 ==>
195             ?E2. WEAK_TRANS E' (label l) E2 /\ (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2)
196         (!E2. TRANS E' (label l) E2 ==>
197             ?E1. WEAK_TRANS E (label l) E1 /\ (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV) E1 E2)
198         (!E1. TRANS E tau E1 ==> ?E2. EPS E' E2 /\ (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2)
199         (!E2. TRANS E' tau E2 ==> ?E1. EPS E E1 /\ (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV) E1 E2)
200     )
201
202 (* Extracted above definition into smaller pieces for easier uses *)
203 val WEAK_BISIM_UPTO_TRANS_label = store_thm (
204     "WEAK_BISIM_UPTO_TRANS_label",
205     '!Wbsm. WEAK_BISIM_UPTO Wbsm ==>
206     !E E'. Wbsm E E' ==>
207     !l E1. TRANS E (label l) E1 ==>
208     ?E2. WEAK_TRANS E' (label l) E2 /\ (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV

```

```

209     PROVE_TAC [WEAK_BISIM_UPTO]);
210
211   val WEAK_BISIM_UPTO_TRANS_label' = store_thm (
212     "WEAK_BISIM_UPTO_TRANS_label'",
213     '!'Wbsm. WEAK_BISIM_UPTO Wbsm ==>
214       !E E'. Wbsm E E' ==>
215         !l E2. TRANS E' (label l) E2 ==>
216           ?E1. WEAK_TRANS E (label l) E1 /\ (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV
217     PROVE_TAC [WEAK_BISIM_UPTO]);
218
219   val WEAK_BISIM_UPTO_TRANS_tau = store_thm (
220     "WEAK_BISIM_UPTO_TRANS_tau",
221     '!'Wbsm. WEAK_BISIM_UPTO Wbsm ==>
222       !E E'. Wbsm E E' ==>
223         !E1. TRANS E tau E1 ==> ?E2. EPS E' E2 /\ (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV
224     PROVE_TAC [WEAK_BISIM_UPTO]);
225
226   val WEAK_BISIM_UPTO_TRANS_tau' = store_thm (
227     "WEAK_BISIM_UPTO_TRANS_tau'",
228     '!'Wbsm. WEAK_BISIM_UPTO Wbsm ==>
229       !E E'. Wbsm E E' ==>
230         !E2. TRANS E' tau E2 ==> ?E1. EPS E E1 /\ (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV
231     PROVE_TAC [WEAK_BISIM_UPTO]);
232
233   val IDENTITY_WEAK_BISIM_UPTO_lemma = store_thm (
234     "IDENTITY_WEAK_BISIM_UPTO_lemma",
235     '!'E. (WEAK_EQUIV 0 (\x y. x = y) 0 STRONG_EQUIV) E E'',
236     GEN_TAC >> REWRITE_TAC [O_DEF] >> BETA_TAC
237   >> Q.EXISTS_TAC 'E' >> REWRITE_TAC [WEAK_EQUIV_REFL]
238   >> Q.EXISTS_TAC 'E' >> REWRITE_TAC [STRONG_EQUIV_REFL]);
239
240   val IDENTITY_WEAK_BISIM_UPTO_lemma' = store_thm (
241     "IDENTITY_WEAK_BISIM_UPTO_lemma'",
242     '!'E. (STRONG_EQUIV 0 (\x y. x = y) 0 WEAK_EQUIV) E E'',
243     GEN_TAC >> REWRITE_TAC [O_DEF] >> BETA_TAC
244   >> Q.EXISTS_TAC 'E' >> REWRITE_TAC [STRONG_EQUIV_REFL]
245   >> Q.EXISTS_TAC 'E' >> REWRITE_TAC [WEAK_EQUIV_REFL]);
246
247   val IDENTITY_WEAK_BISIM_UPTO = store_thm (
248     "IDENTITY_WEAK_BISIM_UPTO", '!'WEAK_BISIM_UPTO (\x y. x = y)'',
249     PURE_ONCE_REWRITE_TAC [WEAK_BISIM_UPTO]
250   >> BETA_TAC
251   >> REPEAT STRIP_TAC (* 4 sub-goals here *)
252   >| [ (* goal 1 (of 4) *)
253       ASSUME_TAC (REWRITE_RULE [ASSUME '!'E :('a, 'b) CCS = E''']
254         (ASSUME '!'TRANS E (label l) E1'')) \\\
255       IMP_RES_TAC TRANS_IMP_WEAK_TRANS \\\
256       Q.EXISTS_TAC 'E1' >> ASM_REWRITE_TAC [] \\\
257       REWRITE_TAC [IDENTITY_WEAK_BISIM_UPTO_lemma],
258     (* goal 2 (of 4) *)
259       IMP_RES_TAC TRANS_IMP_WEAK_TRANS \\\
260       Q.EXISTS_TAC 'E2' >> ASM_REWRITE_TAC [] \\\
261       REWRITE_TAC [IDENTITY_WEAK_BISIM_UPTO_lemma'],
262     (* goal 3 (of 4) *)
263       ASSUME_TAC (REWRITE_RULE [ASSUME '!'E :('a, 'b) CCS = E''']
264         (ASSUME '!'TRANS E tau E1'')) \\\
265       IMP_RES_TAC ONE_TAU \\\
266       Q.EXISTS_TAC 'E1' >> ASM_REWRITE_TAC [] \\\
267       REWRITE_TAC [IDENTITY_WEAK_BISIM_UPTO_lemma],
268     (* goal 4 (of 4) *)
269       IMP_RES_TAC ONE_TAU \\\

```

```

270     Q.EXISTS_TAC 'E2' >> ASM_REWRITE_TAC [] \\  

271     REWRITE_TAC [IDENTITY_WEAK_BISIM_UPTO_lemma'] ]]);  

272  

273 val CONVERSE_WEAK_BISIM_UPTO_lemma = store_thm (  

274     "CONVERSE_WEAK_BISIM_UPTO_lemma",  

275     '!Wbsm E E'. (WEAK_EQUIV O (\x y. Wbsm y x) O STRONG_EQUIV) E E' =  

276         (STRONG_EQUIV O Wbsm O WEAK_EQUIV) E' E',  

277     rpt GEN_TAC  

278     >> EQ_TAC (* 2 sub-goals here *)  

279     >| [ (* goal 1 (of 2) *)  

280         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

281         'STRONG_EQUIV y' E' by PROVE_TAC [STRONG_EQUIV_SYM] \\  

282         'WEAK_EQUIV E' y' by PROVE_TAC [WEAK_EQUIV_SYM] \\  

283         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \\  

284         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [],  

285         (* goal 2 (of 2) *)  

286         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

287         'STRONG_EQUIV E y' by PROVE_TAC [STRONG_EQUIV_SYM] \\  

288         'WEAK_EQUIV y' E' by PROVE_TAC [WEAK_EQUIV_SYM] \\  

289         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \\  

290         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] ]]);  

291  

292 val CONVERSE_WEAK_BISIM_UPTO_lemma' = store_thm (  

293     "CONVERSE_WEAK_BISIM_UPTO_lemma'",  

294     '!Wbsm E E'. (STRONG_EQUIV O (\x y. Wbsm y x) O WEAK_EQUIV) E E' =  

295         (WEAK_EQUIV O Wbsm O STRONG_EQUIV) E' E',  

296     rpt GEN_TAC  

297     >> EQ_TAC (* 2 sub-goals here *)  

298     >| [ (* goal 1 (of 2) *)  

299         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

300         'STRONG_EQUIV E' y' by PROVE_TAC [STRONG_EQUIV_SYM] \\  

301         'WEAK_EQUIV y' E' by PROVE_TAC [WEAK_EQUIV_SYM] \\  

302         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \\  

303         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [],  

304         (* goal 2 (of 2) *)  

305         REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

306         'STRONG_EQUIV y' E' by PROVE_TAC [STRONG_EQUIV_SYM] \\  

307         'WEAK_EQUIV E y' by PROVE_TAC [WEAK_EQUIV_SYM] \\  

308         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] \\  

309         Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC [] ]]);  

310  

311 val CONVERSE_WEAK_BISIM_UPTO = store_thm (  

312     "CONVERSE_WEAK_BISIM_UPTO",  

313     '!Wbsm. WEAK_BISIM_UPTO Wbsm ==> WEAK_BISIM_UPTO (\x y. Wbsm y x)',  

314     GEN_TAC  

315     >> PURE_ONCE_REWRITE_TAC [WEAK_BISIM_UPTO]  

316     >> BETA_TAC  

317     >> rpt STRIP_TAC  

318     >> RES_TAC (* 4 sub-goals here *)  

319     >| [ (* goal 1 (of 4) *)  

320         Q.EXISTS_TAC 'E1' >> ASM_REWRITE_TAC [] \\  

321         REWRITE_TAC [CONVERSE_WEAK_BISIM_UPTO_lemma] \\  

322         ASM_REWRITE_TAC [],  

323         (* goal 2 (of 4) *)  

324         Q.EXISTS_TAC 'E2' >> ASM_REWRITE_TAC [] \\  

325         REWRITE_TAC [CONVERSE_WEAK_BISIM_UPTO_lemma'] \\  

326         ASM_REWRITE_TAC [],  

327         (* goal 3 (of 4) *)  

328         Q.EXISTS_TAC 'E1' >> ASM_REWRITE_TAC [] \\  

329         REWRITE_TAC [CONVERSE_WEAK_BISIM_UPTO_lemma] \\  

330         ASM_REWRITE_TAC [],

```

```

331      (* goal 4 (of 4) *)
332      Q.EXISTS_TAC 'E2' >> ASM_REWRITE_TAC [] \\  

333      REWRITE_TAC [CONVERSE_WEAK_BISIM_UPTO_lemma'] \\  

334      ASM_REWRITE_TAC [] ]);
335
336 val WEAK_BISIM_UPTO_EPS = store_thm ((* NEW *)
337   "WEAK_BISIM_UPTO_EPS",
338   '!'Wbsm. WEAK_BISIM_UPTO Wbsm ==>
339     !E E'. Wbsm E E' ==>
340     !E1. EPS E E1 ==> ?E2. EPS E' E2 /\ (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2
341   rpt STRIP_TAC
342   >> PAT_X_ASSUM 'WEAK_BISIM_UPTO Wbsm' MP_TAC
343   >> Q.PAT_X_ASSUM 'Wbsm E E' MP_TAC
344   >> POP_ASSUM MP_TAC
345   >> Q.SPEC_TAC ('E1', 'E1')
346   >> Q.SPEC_TAC ('E', 'E')
347   >> HO_MATCH_MP_TAC EPS_ind_right (* must use right induct here! *)
348   >> rpt STRIP_TAC (* 2 sub-goals here *)
349   >| [ (* goal 1 (of 2) *)
350     Q.EXISTS_TAC 'E' \\  

351     RW_TAC std_ss [EPS_REFL] \\  

352     REWRITE_TAC [O_DEF] >> BETA_TAC \\  

353     Q.EXISTS_TAC 'E' >> REWRITE_TAC [WEAK_EQUIV_REFL] \\  

354     Q.EXISTS_TAC 'E' >> ASM_REWRITE_TAC [STRONG_EQUIV_REFL],
355     (* goal 2 (of 2) *)
356     RES_TAC \\  

357     POP_ASSUM (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\  

358     STRIP_ASSUME_TAC (ONCE_REWRITE_RULE [PROPERTY_STAR]
359       (ASSUME 'STRONG_EQUIV E1 y')) \\  

360     RES_TAC \\  

361     NTAC 2 (POP_ASSUM K_TAC) \\  

362     STRIP_ASSUME_TAC (REWRITE_RULE [WEAK_BISIM_UPTO]
363       (ASSUME 'WEAK_BISIM_UPTO Wbsm')) \\  

364     POP_ASSUM (STRIP_ASSUME_TAC o (Q.SPECL ['y', 'y'])) \\  

365     RES_TAC \\  

366     NTAC 7 (POP_ASSUM K_TAC) \\  

367     Q.PAT_X_ASSUM 'Wbsm y' y ==> X' K_TAC \\  

368     Q.PAT_X_ASSUM '!1 E1. TRANS y (label 1) E1 ==> P' K_TAC \\  

369     Q.PAT_X_ASSUM '!1 E2. TRANS y (label 1) E2 ==> P' K_TAC \\  

370     IMP_RES_TAC WEAK_EQUIV_EPS \\  

371     Q.EXISTS_TAC 'E2'' \\  

372     CONJ_TAC >- IMP_RES_TAC EPS_TRANS \\  

373     Q.PAT_X_ASSUM 'X E2' E2'' MP_TAC \\  

374     REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

375     (* Induct case:
376       E                                Wbsm                                E'
377                                     //
378       ...                            eps
379                                     //
380     E1 ~ y'      Wbsm      y      = ~ E2
381     /      /      \\\      //
382     tau      tau      eps      eps
383     /      /      \\\      //
384     E1' ~ E2' ~ y''' Wbsm y'' = ~ E2'' = ~ E2'''
385   *)
386     'WEAK_EQUIV y'' E2'' by PROVE_TAC [WEAK_EQUIV_TRANS] \\  

387     'STRONG_EQUIV E1' y''' by PROVE_TAC [STRONG_EQUIV_TRANS] \\  

388     Q.EXISTS_TAC 'y''' >> ASM_REWRITE_TAC [] \\  

389     Q.EXISTS_TAC 'y''' >> ASM_REWRITE_TAC [] ]);
390
391 val WEAK_BISIM_UPTO_EPS' = store_thm ((* NEW *)

```

```

392 "WEAK_BISIM_UPTO_EPS",
393 '!Wbsm. WEAK_BISIM_UPTO Wbsm ==>
394   !E E'. Wbsm E E' ==>
395   !E2. EPS E' E2 ==> ?E1. EPS E E1 /\ (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV) E1 E.
396 GEN_TAC >> DISCH_TAC
397 >> POP_ASSUM (ASSUME_TAC o (MATCH_MP CONVERSE_WEAK_BISIM_UPTO))
398 >> IMP_RES_TAC WEAK_BISIM_UPTO_EPS
399 >> POP_ASSUM MP_TAC
400 >> BETA_TAC >> rpt STRIP_TAC
401 >> RES_TAC
402 >> Q.EXISTS_TAC 'E2'' >> ASM_REWRITE_TAC []
403 >> REWRITE_TAC [GSYM CONVERSE_WEAK_BISIM_UPTO_lemma]
404 >> ASM_REWRITE_TAC []);
405
406 (* Proof sketch:
407   E           Wbsm           E'
408   //          //
409   eps         eps
410   //          //
411   E1' ~ y'    Wbsm    y  =~   E2'    (WEAK_BISIM_UPTO_EPS)
412   /    /      //      //
413   /    l      l      //
414   l    /      //      l
415   / ~ E2'' (~ Wbsm =~) E2'' =~ //
416   E2          E2'''' (WEAK_BISIM_UPTO_TRANS_label)
417   // ~ y'''' Wbsm    y'' =~ //
418   eps //      //      eps
419   //  eps     eps     //
420   //  //      //      //
421   E1 ~ E2'5 (~ Wbsm =~) E2'6 =~ E2'7 (WEAK_BISIM_UPTO_EPS)
422 *)
423 val WEAK_BISIM_UPTO_WEAK_TRANS_label = store_thm ((* NEW *)
424 "WEAK_BISIM_UPTO_WEAK_TRANS_label",
425 '!Wbsm. WEAK_BISIM_UPTO Wbsm ==>
426   !E E'. Wbsm E E' ==>
427     !l E1. WEAK_TRANS E (label l) E1 ==>
428       ?E2. WEAK_TRANS E' (label l) E2 /\
429         (WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2'',
430   rpt STRIP_TAC
431 >> IMP_RES_TAC WEAK_TRANS
432 >> IMP_RES_TAC (MATCH_MP WEAK_BISIM_UPTO_EPS (* lemma 1 used here *)
433   (ASSUME ('WEAK_BISIM_UPTO Wbsm')))
434 >> POP_ASSUM (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF]))
435 >> IMP_RES_TAC PROPERTY_STAR_TRANS
436 >> IMP_RES_TAC WEAK_BISIM_UPTO_TRANS_label
437 >> POP_ASSUM K_TAC
438 >> IMP_RES_TAC WEAK_EQUIV_WEAK_TRANS_label
439 >> Know '(WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E2 E2''''
440 >- ( Q.PAT_X_ASSUM 'X E2'' E2'''' MP_TAC \\  

441   REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \\  

442   'STRONG_EQUIV E2 y'''' by PROVE_TAC [STRONG_EQUIV_TRANS] \\  

443   'WEAK_EQUIV y'' E2'''' by PROVE_TAC [WEAK_EQUIV_TRANS] \\  

444   Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] \\  

445   Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] )
446 >> DISCH_TAC
447 >> POP_ASSUM (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF]))
448 >> IMP_RES_TAC (MATCH_MP STRONG_EQUIV_EPS
449   (ASSUME ('STRONG_EQUIV E2 y'''')))
450 >> IMP_RES_TAC (Q.SPECL ['y''', 'y'']
451   (MATCH_MP WEAK_BISIM_UPTO_EPS (* lemma 1 used here *)
452     (ASSUME ('WEAK_BISIM_UPTO Wbsm'))))

```

```

453 >> NTAC 3 (POP_ASSUM K_TAC)
454 >> IMP_RES_TAC (MATCH_MP WEAK_EQUIV_EPS
455               (ASSUME 'WEAK_EQUIV y' E2'''))
456 >> Know '(WEAK_EQUIV 0 Wbsm 0 STRONG_EQUIV) E1 E2''''''
457 >- ( Q.PAT_X_ASSUM 'X E2'''''' E2'''''' MP_TAC \
458     REWRITE_TAC [O_DEF] >> BETA_TAC >> rpt STRIP_TAC \
459     'STRONG_EQUIV E1 y'''''' by PROVE_TAC [STRONG_EQUIV_TRANS] \
460     'WEAK_EQUIV y'''''' E2'''''' by PROVE_TAC [WEAK_EQUIV_TRANS] \
461     Q.EXISTS_TAC 'y'''''' >> ASM_REWRITE_TAC [] \
462     Q.EXISTS_TAC 'y'''''' >> ASM_REWRITE_TAC [] )
463 >> DISCH_TAC
464 >> Q.EXISTS_TAC 'E2''''''
465 >> ASM_REWRITE_TAC []
466 >> MATCH_MP_TAC EPS_AND_WEAK
467 >> take ['E2', 'E2''']
468 >> ASM_REWRITE_TAC []);
469
470 val WEAK_BISIM_UPTO_WEAK_TRANS_label' = store_thm ((* NEW *)
471 "WEAK_BISIM_UPTO_WEAK_TRANS_label",
472 '!Wbsm. WEAK_BISIM_UPTO Wbsm ==>
473   !E E'. Wbsm E E' ==>
474     !l E2. WEAK_TRANS E' (label l) E2 ==>
475       ?E1. WEAK_TRANS E (label l) E1 /\
476         (STRONG_EQUIV 0 Wbsm 0 WEAK_EQUIV) E1 E2'',
477   GEN_TAC >> DISCH_TAC
478 >> POP_ASSUM (ASSUME_TAC o (MATCH_MP CONVERSE_WEAK_BISIM_UPTO))
479 >> IMP_RES_TAC WEAK_BISIM_UPTO_WEAK_TRANS_label
480 >> POP_ASSUM MP_TAC
481 >> BETA_TAC >> rpt STRIP_TAC
482 >> RES_TAC
483 >> Q.EXISTS_TAC 'E2'' >> ASM_REWRITE_TAC []
484 >> REWRITE_TAC [GSYM CONVERSE_WEAK_BISIM_UPTO_lemma]
485 >> ASM_REWRITE_TAC []);
486
487 (* If S is a bisimulation up to WEAK_EQUIV, then (WEAK_EQUIV 0 S 0 WEAK_EQUIV) is
488    a weak bisimulation. *)
489 val WEAK_BISIM_UPTO_LEMMA = store_thm (
490 "WEAK_BISIM_UPTO_LEMMA",
491 '!Wbsm. WEAK_BISIM_UPTO Wbsm ==> WEAK_BISIM (WEAK_EQUIV 0 Wbsm 0 WEAK_EQUIV)',
492 GEN_TAC
493 >> REWRITE_TAC [WEAK_BISIM, O_DEF]
494 >> rpt STRIP_TAC (* 4 sub-goals here *)
495 >| [ (* goal 1 (of 4) *)
496     IMP_RES_TAC (MATCH_MP WEAK_EQUIV_TRANS_label (ASSUME 'WEAK_EQUIV E y'')) \
497     IMP_RES_TAC (MATCH_MP WEAK_BISIM_UPTO_WEAK_TRANS_label
498                   (ASSUME 'WEAK_BISIM_UPTO Wbsm')) \
499     IMP_RES_TAC (MATCH_MP WEAK_EQUIV_WEAK_TRANS_label
500                   (ASSUME 'WEAK_EQUIV y E'')) \
501     Q.EXISTS_TAC 'E2'' >> ASM_REWRITE_TAC [] \
502     Q.PAT_X_ASSUM 'X E2 E2'' (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \
503     (***)
504         E      ~ =   y'      Wbsm      y      ~ =   E'
505         /      //      \      \      //
506         !l      l      l      l      l
507         /      //      \      \      //
508         E1 ~ = E2 ~ y'' Wbsm y'' ~ = E2' ~ = E2''
509     ***)
510     'WEAK_EQUIV E2 y'''' by PROVE_TAC [STRONG_IMP_WEAK_EQUIV] \
511     'WEAK_EQUIV E1 y'''' by PROVE_TAC [WEAK_EQUIV_TRANS] \
512     'WEAK_EQUIV y'' E2'''' by PROVE_TAC [WEAK_EQUIV_TRANS] \
513     Q.EXISTS_TAC 'y'''' >> ASM_REWRITE_TAC [] \

```

```

514 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [],
515 (* goal 2 (of 4) *)
516 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_TRANS_label' (ASSUME 'WEAK_EQUIV y E'')) \\
517 IMP_RES_TAC (MATCH_MP WEAK_BISIM_UPTO_WEAK_TRANS_label'
518 (ASSUME 'WEAK_BISIM_UPTO Wbsm')) \\
519 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_WEAK_TRANS_label'
520 (ASSUME 'WEAK_EQUIV E y'')) \\
521 Q.EXISTS_TAC 'E1'' >> ASM_REWRITE_TAC [] \\
522 Q.PAT_X_ASSUM 'X E1' E1' (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\
523 (***)
524      E      ~ =      y'      Wbsm      y      ~ =      E'
525      //      //      \\      /
526      l      l      l      l
527      //      //      \\      /
528      E1'' ~ = E1' ~ = y'' Wbsm y'' ~ E1 ~ = E2
529 (***)
530 'WEAK_EQUIV E1'' y'' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
531 'WEAK_EQUIV y'' E1' by PROVE_TAC [STRONG_IMP_WEAK_EQUIV] \\
532 'WEAK_EQUIV y'' E2' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
533 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] \\
534 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [],
535 (* goal 3 (of 4) *)
536 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_TRANS_tau (ASSUME 'WEAK_EQUIV E y'')) \\
537 IMP_RES_TAC (MATCH_MP WEAK_BISIM_UPTO_EPS (ASSUME 'WEAK_BISIM_UPTO Wbsm')) \\
538 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_EPS (ASSUME 'WEAK_EQUIV y E'')) \\
539 Q.EXISTS_TAC 'E2'' >> ASM_REWRITE_TAC [] \\
540 Q.PAT_X_ASSUM 'X E2 E2'' (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\
541 (***)
542      E      ~ =      y'      Wbsm      y      ~ =      E'
543      /      //      \\      //
544      tau      eps      eps      eps
545      /      //      \\      //
546      E1 ~ = E2 ~ y'' Wbsm y'' ~ E2' ~ = E2''
547 (***)
548 'WEAK_EQUIV E2 y'' by PROVE_TAC [STRONG_IMP_WEAK_EQUIV] \\
549 'WEAK_EQUIV E1 y'' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
550 'WEAK_EQUIV y'' E2'' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
551 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] \\
552 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [],
553 (* goal 4 (of 4) *)
554 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_TRANS_tau' (ASSUME 'WEAK_EQUIV y E'')) \\
555 IMP_RES_TAC (MATCH_MP WEAK_BISIM_UPTO_EPS' (ASSUME 'WEAK_BISIM_UPTO Wbsm')) \\
556 IMP_RES_TAC (MATCH_MP WEAK_EQUIV_EPS' (ASSUME 'WEAK_EQUIV E y'')) \\
557 Q.EXISTS_TAC 'E1'' >> ASM_REWRITE_TAC [] \\
558 Q.PAT_X_ASSUM 'X E1' E1' (STRIP_ASSUME_TAC o (REWRITE_RULE [O_DEF])) \\
559 (***)
560      E      ~ =      y'      Wbsm      y      ~ =      E'
561      //      //      \\      /
562      eps      eps      eps      tau
563      //      //      \\      /
564      E1'' ~ = E1' ~ = y'' Wbsm y'' ~ E1 ~ = E2
565 (***)
566 'WEAK_EQUIV E1'' y'' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
567 'WEAK_EQUIV y'' E1' by PROVE_TAC [STRONG_IMP_WEAK_EQUIV] \\
568 'WEAK_EQUIV y'' E2' by PROVE_TAC [WEAK_EQUIV_TRANS] \\
569 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] \\
570 Q.EXISTS_TAC 'y'' >> ASM_REWRITE_TAC [] ];
571
572 val WEAK_BISIM_UPTO_THM = store_thm (
573   "WEAK_BISIM_UPTO_THM",
574   '!Wbsm. WEAK_BISIM_UPTO Wbsm ==> Wbsm RSUBSET WEAK_EQUIV',

```



```

575     rpt STRIP_TAC
576 >> IMP_RES_TAC WEAK_BISIM_UPTO_LEMMA
577 >> IMP_RES_TAC WEAK_BISIM_SUBSET_WEAK_EQUIV
578 >> Suff 'Wbsm RSUBSET (WEAK_EQUIV O Wbsm O WEAK_EQUIV)'
579 >- ( DISCH_TAC \\  

580     Know 'transitive ((RSUBSET) :('a, 'b) simulation -> ('a, 'b) simulation -> bool)'
581     >- PROVE_TAC [RSUBSET_WeakOrder, WeakOrder] \\  

582     RW_TAC std_ss [transitive_def] >> RES_TAC )
583 >> KILL_TAC
584 >> REWRITE_TAC [RSUBSET, O_DEF]
585 >> rpt STRIP_TAC
586 >> 'WEAK_EQUIV x x /\ WEAK_EQUIV y y' by PROVE_TAC [WEAK_EQUIV_REFL]
587 >> Q.EXISTS_TAC 'y' >> ASM_REWRITE_TAC []
588 >> Q.EXISTS_TAC 'x' >> ASM_REWRITE_TAC []
589 (* Hence, to prove  $P \sim Q$ , we only have to find a strong bisimulation up to  $\sim$ 
590    which contains  $(P, Q)$  *));
591
592 (* Bibliography:
593  *
594  * [1] Milner, R.: Communication and concurrency. (1989).
595  * [2] Gorrieri, R., Versari, C.: Introduction to Concurrency Theory. Springer, Cham (2015)
596  * [3] Sangiorgi, D.: Introduction to Bisimulation and Coinduction. Cambridge University Press (2011)
597  * [4] Sangiorgi, D., Rutten, J.: Advanced Topics in Bisimulation and Coinduction.
598      Cambridge University Press (2011).
599  *)
600
601 val _ = export_theory ();
602 val _ = print_theory_to_file "-" "BisimulationUpto.lst";
603 val _ = DB.html_theory "BisimulationUpto";
604
605 (* last updated: Aug 5, 2017 *)

```