Bisimulation

R.J. van Glabbeek
NICTA, Sydney, Australia.
School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia.
Computer Science Department, Stanford University, CA 94305-9045, USA
http://theory.stanford.edu/~rvg rvg@cs.stanford.edu

Bisimulation equivalence is a semantic equivalence relation on labelled transition systems, which are used to represent distributed systems. It identifies systems with the same branching structure.

Labelled transition systems

A labelled transition system consists of a collection of states and a collection of transitions between them. The transitions are labelled by actions from a given set A that happen when the transition is taken, and the states may be labelled by predicates from a given set P that hold in that state.

Definition 1 Let A and P be sets (of *actions* and *predicates*, respectively).
A *labelled transition system* (LTS) over A and P is a triple $(S, \rightarrow, \models)$ with

- S a class (of *states*),
- $\rightarrow$ a collection of binary relations $\xrightarrow{a} \subseteq S \times S$—one for every $a \in A$—(the *transitions*), such that for all $s \in S$ the class $\{t \in S \mid s \xrightarrow{a} t\}$ is a set,
- and $\models \subseteq S \times P$. $s \models p$ says that predicate $p \in P$ *holds* in state $s \in S$.

LTSs with A a singleton (i.e. with $\rightarrow$ a single binary relation on S) are known as *Kripke structures*, the models of modal logic. General LTSs (with A arbitrary) are the Kripke models for polymodal logic. The name "labelled transition system" is employed in concurrency theory. There, the elements of S represent the systems one is interested in, and $s \xrightarrow{a} t$ means that system s can evolve into system t while performing the action a. This approach identifies states and systems: the states of a system s are the systems reachable from s by following the transitions. In this realm P is often taken to be empty, or it contains a single predicate $\surd$ indicating successful termination.

Definition 2 A *process graph* over A and P is a tuple $g = (S, I, \rightarrow, \models)$ with $(S, \rightarrow, \models)$ an LTS over A and P in which S is a set, and $I \in S$.

Process graphs are used in concurrency theory to disambiguate between states and systems. A process graph $(S, I, \rightarrow, \models)$ represents a single system, with S the set of its states and I its initial state. In the context of an LTS $(S, \rightarrow, \models)$ two concurrent systems are modelled by two members of S; in the context of process graphs, they are two different graphs. The *nondeterministic finite automata* used in *automata theory* are process graphs with a finite set of states over a finite alphabet A and a set P consisting of a single predicate denoting *acceptance*.

Written August 2000 for the forgotten *Encyclopedia of Distributed Computing* (J.E. Urban & P. Dasgupta, eds.). "Further reading" added November 2010. To appear in the *Encyclopedia of Parallel Computing* (D. Padua, ed.), Springer, 2011.

Bisimulation equivalence

Bisimulation equivalence is defined on the states of a given LTS, or between different process graphs.

Definition 3 Let $(S, \rightarrow, \models)$ be an LTS over A and P. A *bisimulation* is a binary relation $R \subseteq S \times S$, satisfying:

$\wedge$ if sRt then $s \models p \Leftrightarrow t \models p$ for all $p \in P$,

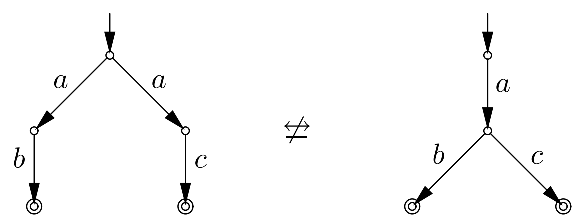
$\wedge$ if sRt and $s \stackrel{a}{\longrightarrow} s'$ with $a \in A$, then there exists a t' with $t \stackrel{a}{\longrightarrow} t'$ and $s'Rt'$,

$\wedge$ if sRt and $t \stackrel{a}{\longrightarrow} t'$ with $a \in A$, then there exists an s' with $s \stackrel{a}{\longrightarrow} s'$ and $s'Rt'$.

Two states $s, t \in S$ are *bisimilar*, denoted $s \mathrel{\underline{\leftrightarrow}} t$, if there exists a bisimulation R with sRt.

Bisimilarity turns out to be an equivalence relation on S, and is also called *bisimulation equivalence*.

Definition 4 Let $g = (S, I, \rightarrow, \models)$ and $h = (S', I', \rightarrow', \models')$ be process graphs over A and P. A *bisimulation* between g and h is a binary relation $R \subseteq S \times S'$, satisfying IRI' and the same three clauses as above. g and h are *bisimilar*, denoted $g \mathrel{\underline{\leftrightarrow}} h$, if there exists a bisimulation between them.



Example The two process graphs above (over $A = \{a, b, c\}$ and $P = \{\surd\}$), in which the initial states are indicated by short incoming arrows and the final states (the ones labelled with $\surd$) by double circles, are not bisimulation equivalent, even though in automata theory they accept the same language. The choice between b and c is made at a different moment (namely before vs. after the a-action); i.e. the two systems have a different *branching structure*. Bisimulation semantics distinguishes systems that differ in this manner.

Modal logic

(Poly)modal logic is an extension of propositional logic with formulas $\langle a \rangle \varphi$, saying that it is possible to follow an a-transition after which the formula φ holds. Modal formulas are interpreted on the states of labelled transition systems. Two systems are bisimilar iff they satisfy the same infinitary modal formulas.

Definition 5 The language $\mathcal{L}$ of *polymodal logic* over A and P is given by:

- $\top \in \mathcal{L}$,
- $p \in \mathcal{L}$ for all $p \in P$,
- if $\varphi, \psi \in \mathcal{L}$ for then $\varphi \wedge \psi \in \mathcal{L}$,
- if $\varphi \in \mathcal{L}$ then $\neg\varphi \in \mathcal{L}$,
- if $\varphi \in \mathcal{L}$ and $a \in A$ then $\langle a \rangle \varphi \in \mathcal{L}$.

Basic (as opposed to *poly-*) modal logic is the special case where $|A| = 1$; there $\langle a\rangle\varphi$ is simply denoted $\diamond\varphi$. The *Hennessy-Milner logic* is polymodal logic with $P = \emptyset$. The language $\mathcal{L}^\infty$ of *infinitary polymodal logic* over A and P is obtained from $\mathcal{L}$ by additionally allowing $\bigwedge_{i\in I}\varphi_i$ to be in $\mathcal{L}^\infty$ for arbitrary index sets I and $\varphi_i \in \mathcal{L}^\infty$ for $i \in I$. The connectives $\top$ and $\wedge$ are then the special cases $I = \emptyset$ and $|I| = 2$.

Definition 6 Let $(S, \rightarrow, \models)$ be an LTS over A and P. The relation $\models\, \subseteq S \times P$ can be extended to the *satisfaction relation* $\models\, \subseteq S \times \mathcal{L}^\infty$, by defining

- $s \models \bigwedge_{i\in I}\varphi_i$ if $s \models \varphi_i$ for all $i \in I$—in particular, $s \models \top$ for any state $s \in S$,
- $s \models \neg\varphi$ if $s \not\models \varphi$,
- $s \models \langle a\rangle\varphi$ if there is a state t with $s \xrightarrow{a} t$ and $t \models \varphi$.

Write $\mathcal{L}(s)$ for $\{\varphi \in \mathcal{L} \mid s \models \varphi\}$.

Theorem 1 [5] Let $(S, \rightarrow, \models)$ be an LTS and $s, t \in S$. Then $s \leftrightarroweq t \Leftrightarrow \mathcal{L}^\infty(s) = \mathcal{L}^\infty(t)$.

In case the systems s and t are image finite, it suffices to consider finitary polymodal formulas only [3]. In fact, for this purpose it is enough to require that one of s and t is image finite.

Definition 7 Let $(S, \rightarrow, \models)$ be an LTS. A state $t \in S$ is *reachable* from $s \in S$ if there are $s_i \in S$ and $a_i \in A$ for $i = 0, ..., n$ with $s = s_0$, $s_{i-1} \xrightarrow{a_i} s_i$ for $i = 1, ..., n$, and $s_n = t$. A state $s \in S$ is *image finite* if for every state $t \in S$ reachable from s and for every $a \in A$, the set $\{u \in S \mid t \xrightarrow{a} u\}$ is finite.

Theorem 2 [4] Let $(S, \rightarrow, \models)$ be an LTS and $s, t \in S$ with s image finite. Then $s \leftrightarroweq t \Leftrightarrow \mathcal{L}(s) = \mathcal{L}(t)$.

Non-well-founded sets

Another characterization of bisimulation semantics can be given by means of Aczel's universe $\mathcal{V}$ of non-well-founded sets [1]. This universe is an extension of the Von Neumann universe of well-founded sets, where the axiom of foundation (every chain $x_0 \ni x_1 \ni \cdots$ terminates) is replaced by an *anti-foundation axiom*.

Definition 8 Let $(S, \rightarrow, \models)$ be an LTS, and let $\mathcal{B}$ denote the unique function $\mathcal{M} : S \rightarrow \mathcal{V}$ satisfying, for all $s \in S$,

$$\mathcal{M}(s) = \{\langle a, \mathcal{M}(t)\rangle \mid s \xrightarrow{a} t\}.$$

It follows from Aczel's anti-foundation axiom that such a function exists. In fact, the axiom amounts to saying that systems of equations like the one above have unique solutions. $\mathcal{B}(s)$ could be taken to be the *branching structure* of s. The following theorem then says that two systems are bisimilar iff they have the same branching structure.

Theorem 3 [2] Let $(S, \rightarrow, \models)$ be an LTS and $s, t \in S$. Then $s \leftrightarroweq t \;\Leftrightarrow\; \mathcal{B}(s) = \mathcal{B}(t)$.

Abstraction

In concurrency theory it is often useful to distinguish between *internal actions*, that do not admit interactions with the outside world, and *external* ones. As normally there is no need to distinguish the internal actions from each other, they all have the same name, namely τ. If A is the set of external actions a certain class of systems may perform, then $A_\tau := A \dot\cup \{\tau\}$. Systems in that class are then represented by labelled transition systems over A_τ and a set of predicates P. The variant of bisimulation equivalence that treats τ just like any action of A is called *strong bisimulation equivalence*. Often, however, one wants to abstract from internal actions to various degrees. A system doing two τ actions in succession is then considered equivalent to a system doing just one. However, a system that can do either a or b is considered different from a system that can do either a or first τ and then b, because if the former system is placed in an environment where b cannot happen, it can still do a instead, whereas the latter system may reach a state (by executing the τ action) in which a is no longer possible.

Several versions of bisimulation equivalence that formalize these desiderata occur in the literature. *Branching bisimulation equivalence* [2], like strong bisimulation, faithfully preserves the branching structure of related systems. The notions of *weak* and *delay* bisimulation equivalence, which were both introduced by Milner under the name *observational equivalence*, make more identifications, motivated by observable machine-behavior according to certain testing scenarios.

Write $s \Longrightarrow t$ for $\exists n \geq 0 : \exists s_0, ..., s_n : s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n = t$, i.e. a (possibly empty) path of τ-steps from s to t. Furthermore, for $a \in A_\tau$, write $s \xrightarrow{(a)} t$ for $s \xrightarrow{a} t \vee (a = \tau \wedge s = t)$. Thus $\xrightarrow{(a)}$ is the same as $\xrightarrow{a}$ for $a \in A$, and $\xrightarrow{(\tau)}$ denotes zero or one τ-steps.

Definition 9 Let $(S, \rightarrow, \models)$ be an LTS over A_τ and P. Two states $s, t \in S$ are *branching bisimulation equivalent*, denoted $s \leftrightarroweq_b t$, if they are related by a binary relation $R \subseteq S \times S$ (a *branching bisimulation*), satisfying:

$\wedge$ if sRt and $s \models p$ with $p \in P$, then there is a t_1 with $t \Longrightarrow t_1 \models p$ and sRt_1,

$\wedge$ if sRt and $t \models p$ with $p \in P$, then there is a s_1 with $s \Longrightarrow s_1 \models p$ and s_1Rt,

$\wedge$ if sRt and $s \xrightarrow{a} s'$ with $a \in A_\tau$, then there are t_1, t_2, t' with $t \Longrightarrow t_1 \xrightarrow{(a)} t_2 = t'$, sRt_1 and $s'Rt'$,

$\wedge$ if sRt and $t \xrightarrow{a} t'$ with $a \in A_\tau$, then there are s_1, s_2, s' with $s \Longrightarrow s_1 \xrightarrow{(a)} s_2 = s'$, s_1Rt and $s'Rt'$.

Delay bisimulation equivalence, $\leftrightarroweq_d$, is obtained by dropping the requirements sRt_1 and s_1Rt. *Weak bisimulation equivalence* [5], $\leftrightarroweq_w$, is obtained by furthermore relaxing the requirements $t_2 = t'$ and $s_2 = s'$ to $t_2 \Longrightarrow t'$ and $s_2 \Longrightarrow s'$.

These definition stem from concurrency theory. On Kripke structures, when studying modal or temporal logics, normally a stronger version of the first two conditions is imposed:

$\wedge$ if sRt and $p \in P$, then $s \models p \Leftrightarrow t \models p$.

For systems without τ's all these notions coincide with strong bisimulation equivalence.

Concurrency

When applied to *parallel systems*, capable of performing different actions at the same time, the versions of bisimulation discussed here employ *interleaving semantics*: no distinction is made between true parallelism and its nondeterministic sequential simulation. Versions of bisimulation that do make such a distinction have been developed as well, most notably the *ST-bisimulation* [2], that takes temporal overlap of actions into account, and the *history preserving bisimulation* [2] that even keeps track of causal relations between actions. For this purpose, system representations such as *Petri nets* or *event structures* are often used instead of labelled transition systems.

References

[1] P. Aczel (1988): *Non-well-founded Sets*, *CSLI Lecture Notes* 14. Stanford University.

[2] R.J. van Glabbeek (1990): *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Free University, Amsterdam. Second edition available as CWI tract 109, CWI, Amsterdam 1996.

[3] M. Hennessy & R. Milner (1985): *Algebraic laws for nondeterminism and concurrency. Journal of the ACM* 32(1), pp. 137–161.

[4] M.J. Hollenberg (1995): *Hennessy-Milner classes and process algebra.* In A. Ponse, M. de Rijke & Y. Venema, editors: *Modal Logic and Process Algebra: a Bisimulation Perspective*, *CSLI Lecture Notes* 53, CSLI Publications, Stanford, California, pp. 187–216.

[5] R. Milner (1990): *Operational and algebraic semantics of concurrent processes.* In J. van Leeuwen, editor: *Handbook of Theoretical Computer Science*, chapter 19, Elsevier Science Publishers B.V. (North-Holland), pp. 1201–1242.

Further reading

Gentle introductions to bisimulation semantics, with many examples of applications, can be found in the textbooks:

– J.C.M. Baeten & W.P. Weijland (1990): *Process Algebra*, Cambridge University Press.

– R. Milner (1989): *Communication and Concurrency*, Prentice Hall.

An historical perspective on bisimulation appears in

– D. Sangiorgi (2009): *On the origins of bisimulation and coinduction*, *ACM Transactions on Programming Languages and Systems* 31(4).