A Characterisation of Weak Bisimulation Congruence

Rob J. van Glabbeek

National ICT Australia
and School of Computer Science and Engineering,
The University of New South Wales
rvg@cs.stanford.edu

Abstract. This paper shows that weak bisimulation congruence can be characterised as rooted weak bisimulation equivalence, even without making assumptions on the cardinality of the sets of states or actions of the processes under consideration.

Introduction

Weak bisimulation equivalence, also known as observation equivalence [7], is a fundamental semantic equivalence used in system verification, and one of the first proposed in the literature. It upgrades strong bisimulation equivalence by featuring abstraction from internal actions.

In order to allow compositional system verification, semantic equivalence relations need to be *congruences* for the operators under consideration, meaning that the equivalence class of an n-ary operator f applied to arguments $p_1, \ldots, p_n$ is completely determined by the equivalence classes of these arguments. Although strong bisimulation equivalence is a congruence for the operators of CCS, ACP_τ and many other languages found in the literature, weak bisimulation equivalence fails to be a congruence for the *choice* or *alternative composition* operator $+$ of CCS, as well as for the left-merge $\between$ of ACP_τ. To bypass this problem, one uses the coarsest congruence relation for $+$ that is finer than weak bisimulation equivalence, called *weak bisimulation congruence*, and characterised as *rooted weak bisimulation equivalence* in [2]. This equivalence turns out to be a minor variant of weak bisimulation equivalence, and a congruence for all of CCS, ACP_τ and many other languages.

Classical proof sketches arguing that rooted weak bisimulation equivalence is indeed weak bisimulation congruence typically make some cardinally assumptions, such as that there is an infinite alphabet of actions of which each process uses only a finite subset. The current contribution establishes the validity of this characterisation without making such assumptions. It also argues that the *root condition* that turns weak bisimulation into rooted weak bisimulation embodies two properties, one of which is needed to obtain a congruence for the $+$, and one to obtain a congruence for the left-merge.

A. Middeldorp et al. (Eds.): Processes... (Klop Festschrift), LNCS 3838, pp. 26–39, 2005.

1 Process Graphs

Definition 1 ([3]). A *process graph* over an alphabet of actions Act is a rooted, directed graph whose edges are labelled by elements of Act. Formally, a process graph g is a triple $(\text{NODES}(g), \text{ROOT}(g), \text{EDGES}(g))$, where

- $\text{NODES}(g)$ is a set, of which the elements are called the *nodes* or *states* of g,
- $\text{ROOT}(g) \in \text{NODES}(g)$ is a special node: the *root* or *initial state* of g,
- and $\text{EDGES}(g) \subseteq \text{NODES}(g) \times Act \times \text{NODES}(g)$ is a set of triples (s, a, t) with $s, t \in \text{NODES}(g)$ and $a \in Act$: the *edges* or *transitions* of g.

Normally, one is not interested in the names of the nodes in a process graph. For this reason, process graphs are considered up to isomorphism.

Definition 2. Let g and h be process graphs. A *graph isomorphism* between g and h is a bijective function $f : \text{NODES}(g) \to \text{NODES}(h)$ satisfying

- $f(\text{ROOT}(g)) = \text{ROOT}(h)$ and
- $(s, a, t) \in \text{EDGES}(g) \;\Leftrightarrow\; (f(s), a, f(t)) \in \text{EDGES}(h)$.

Graphs g and h are *isomorphic*, notation $g \cong h$, if there exists a graph isomorphism between them.

If $g \cong h$ then g and h differ only in the identity of their nodes. Graph isomorphism is an equivalence relation on the class of process graphs.

Further on, process graphs are pictured by using open dots ($\circ$) to denote nodes, and labelled arrows to denote edges. The root is represented by an incoming arrow, not originating from another node. These drawings present process graphs only up to isomorphism.

Let $\mathbb{G}(Act)$ be the class of process graphs over the alphabet of actions Act *up to isomorphism*. This means that I am satisfied with a level of precision in describing elements of $\mathbb{G}(Act)$ that fails to distinguish isomorphic process graphs. In the digression below I will indicate how to raise the precision of my definitions to a fully formal level; the digression should also make clear that it is not really worthwhile to maintain this level throughout the paper.

Next I define the most basic process algebraic operations on $\mathbb{G}(Act)$: a constant 0 for *inaction*, a binary infix written operator $+$ for *alternative composition* or *choice*, and unary operators $a.$ for *action prefixing* for each $a \in Act$. For the sake of convenience, in the definition below I will only consider *root-acyclic* process graphs. In Sect. 3 I will extend the definition to arbitrary process graphs.

Definition 3 ([3]). A process graph is *root-acyclic* if it has no incoming edges at the root. Let $\mathbb{G}^\rho(Act)$ be the class of root-acyclic process graphs over Act up to isomorphism. The constant 0 and the operators $a.$ and $+_\rho$ are defined on $\mathbb{G}^\rho(Act)$ as follows. (The subscript ρ serves to distinguish this alternative composition from the more general one that will be defined in Sect. 3.)

- 0 is interpreted as the trivial graph, having one node (the root) and no edges;
- $a.g$ is obtained from g by adding a new node, which will be the root of $a.g$, and a new a-labelled edge from the root of $a.g$ to the root of g;
- $g +_\rho h$ is obtained by identifying the root nodes of disjoint copies of g and h.