Game Constructions that Are Safe for Bisimulation

Marc Pauly

March 31, 1999

Abstract

The notion of bisimulation is extended to *Game Logic* (*GL*), a logic for reasoning about winning strategies in 2-player games. We show that all game constructions of *GL* are safe for bisimulation, i.e. that an atomic bisimulation can be lifted to non-atomic games constructed by the operations of *GL*. As a consequence, bisimilar states satisfy the same *GL*-formulas.

Contents

1 Introduction

While the work of Johan van Benthem covers many areas related to logic, it seems fair to say that the topic of bisimulation can be identified as one of his long-standing interests. Starting with his Ph.D. thesis [vB76] where he identified the modal fragment of first-order logic as the bisimulation-invariant fragment, his interest in questions related to bisimulation continues to the present day, as witnessed e.g. by [vB97, vB96]. Furthermore, he has managed to convey his enthusiasm for this topic (as for many others) to a number of his Ph.D. students including the present author, resulting in theses such as [d'A98] and [Hol98].

The current paper seems appropriate for this Liber Amicorum not only because it is yet another chapter in the great book on bisimulation, but also because it is inspired directly by a paper of Johan van Benthem entitled *Program Constructions that Are Safe for Bisimulation* [vB93]. In that paper, van Benthem shows that (1) programs constructed from atomic relations and tests by means of composition, union and negation are *safe for bisimulation* and that conversely (2) all first-order definable relations which are safe for bisimulation can be constructed in this way. This result can be seen as an expressive-completeness result: Given that we want to treat bisimilar models/processes as the same, we want our program operations to be safe for bisimulation, i.e. they should in some sense preserve the bisimulation. If we accept that any program construction we may want to define has to satisfy this condition, van Benthem's result says that the operations of composition, union and negation are sufficient to construct any new program operation one may think of, provided it is first-order definable.

As the title of this contribution suggests, the aim of this paper is to extend van Benthem's line of investigation from programs to games. From a game perspective, one can see programs as special kinds of games, namely games where one player (the computer) makes all the moves. From this angle, it is natural to ask what happens if one adds game constructions which allow the other player to make moves as well. A propositional logic of games which extends Propositional Dynamic Logic (*PDL*, [Har84, KT90]) with such interaction is *Game Logic* (*GL*), introduced in [Par85]. After giving a short introduction to *GL* (section 2), we shall show that the notion of bisimulation-safety can be extended to game constructions, and that the game constructions of *GL* are indeed safe for bisimulation. Note that this only establishes one half of the game version of van Benthem's result, the less challenging half to be sure. Still, it should suffice to show that (and how) van Benthem's approach can be lifted to a more general modal logic. The more challenging question as to whether one can similarly find an analogous converse result will be left for some other occasion...

2 Playing Games in Kripke Models

GL is a logic to reason about winning strategies in strictly competitive games between two players who we shall call Angel and Demon. The underlying

semantic model in which games are played is a Kripke model. (This is more restrictive than in the original paper [Par85] but yields a simpler exposition.) An atomic game g played at state s consists of Angel choosing one of the g-successors of s; Angel looses if s has no such successors. A test game $\varphi?$ consists of checking through a neutral arbiter whether proposition φ holds at that state. If it does, nothing happens (i.e. another game can be played) and otherwise, Demon wins. As for the game operations, $\pi_1 \cup \pi_2$ denotes the game where first Angel chooses which of the two subgames to play and then that game is played. The sequential composition $\pi_1; \pi_2$ of two games consists of first playing π_1 and then π_2, and the iterated game π^* has Angel choose how many π games (possibly none) she wants to play.

Reading the informal explanation of these games carefully, one can notice that Angel makes all the choices/moves in these games. In fact, the game operations described so far do not go beyond the program operations known in *PDL*. In order to introduce interaction between the players, *GL* adds an operator *dual* for role interchange: Playing the dual game π^d is the same as playing π with the roles of the players reversed, i.e. any choice made by Angel in π will be made by Demon in π^d and vice versa.

Playing a game consists of moving through the states of the model based on the choices the players make. After a game is played, a certain final state of the game (i.e. state of the model) is reached. We want $\langle\pi\rangle\varphi$ to express that Angel can play π such that in the resulting state φ holds. In more game-theoretic terminology, Angel has a winning strategy for φ in π, a strategy which *brings about* φ. Similarly, $[\pi]\varphi$ denotes the existence of such a winning strategy for Demon.

After having provided some first intuitions for *GL*, we shall define the syntax and semantics of *GL* formally. The language of *GL* consists of two sorts, games and propositions. Given a set of atomic games Π_0 and a set of atomic propositions Φ_0, games π and propositions φ can have the following syntactic forms, yielding the set of games Π and the set of propositions/formulas Φ:

$$\begin{array}{ll} \pi := & g \mid \varphi? \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^d \\ \varphi := & \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \end{array}$$

where $p \in \Phi_0$ and $g \in \Pi_0$. As usual, we define $\top := \neg\bot$, $[\pi]\varphi := \neg\langle\pi\rangle\neg\varphi$, $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$ and $\varphi \to \psi := \neg\varphi \vee \psi$.

As for the semantics, a *Kripke model* $\mathcal{I} = (S, \{r_a | a \in \Pi_0\}, V)$, consists of a set of states S, a valuation for the propositional letters V such that for all $p \in \Phi_0$ we have $V(p) \subseteq S$, and a set of relations $r_a \subseteq S \times S$. Given such a model, we can define inductively when a formula φ is true in $\mathcal{I}$ at state s, denoted as $\mathcal{I}, s \models \varphi$:

$$\begin{array}{lll} \mathcal{I}, s \not\models \bot & & \\ \mathcal{I}, s \models p & \text{iff } p \in \Phi_0 \text{ and } s \in V(p) & \\ \mathcal{I}, s \models \neg\varphi & \text{iff } \mathcal{I}, s \not\models \varphi & \\ \mathcal{I}, s \models \varphi \vee \psi & \text{iff } \mathcal{I}, s \models \varphi \text{ or } \mathcal{I}, s \models \psi & \\ \mathcal{I}, s \models \langle\pi\rangle\varphi & \text{iff } s \in R_\pi(\varphi^{\mathcal{I}}) & \end{array}$$

where $\varphi^{\mathcal{I}} := \{s \in S | \mathcal{I}, s \models \varphi\}$. The modal clause of the definition makes use of the function $R_\pi : \mathcal{P}(S) \to \mathcal{P}(S)$. Given a set of states X, $R_\pi(X)$ denotes the set of states from which Angel has a strategy in π to bring about X. The function is defined by induction on π for any $Y \subseteq S$:

$$\begin{array}{rl} R_g(Y) & = \{s \in S | \exists t : s r_g t \ \&\ t \in Y\} \ \text{ for } g \in \Pi_0 \\ R_{\alpha;\beta}(Y) & = R_\alpha(R_\beta(Y)) \\ R_{\alpha\cup\beta}(Y) & = R_\alpha(Y) \cup R_\beta(Y) \\ R_{\varphi?}(Y) & = \varphi^{\mathcal{I}} \cap Y \\ R_{\alpha^d}(Y) & = \overline{R_\alpha(\overline{Y})} \\ R_{\alpha^*}(Y) & = \mu X.(Y \cup R_\alpha(X) \subseteq X) \end{array}$$

In this definition, $\mu X.f(X)$ denotes the smallest set X satisfying $f(X)$. Note that in order to guarantee the existence of this least fixpoint, R_π should be monotonic for every game π, i.e. if $X \subseteq Y$ then $R_\pi(X) \subseteq R_\pi(Y)$. We shall not carry out the proof that this is indeed the case, but only remark that the monotonicity requirement is a natural one given our interpretation in terms of strategies: If Angel has a strategy to bring about a state in X, then that strategy trivially brings about a state in Y for every $Y \supseteq X$.

3 Generalizing Bisimulation for Games

Bisimulation provides an answer to the question: *when should two models or processes be considered the same?* Different criteria may come to mind depending on what aspects of models one is interested in. If only interested in observable properties of processes, one may choose for finite-trace equivalence, but if interested in mathematical structure, one may choose isomorphism. These equivalence notions (see e.g. [vBB93] for an overview) partition the class of models into equivalence classes, and one may order equivalence notions according to how fine-grained the partition is which they induce. While finite-trace equivalence is often considered as too coarse and isomorphism as too fine, bisimulation is situated between these two extremes.

Definition 1 (Bisimulation) *Let $\mathcal{I} = (S, \{r_a | a \in \Pi_0\}, V)$ and $\mathcal{I}' = (S', \{r'_a | a \in \Pi_0\}, V')$ be two Kripke models. Then $\sim \subseteq S \times S'$ is a bisimulation between $\mathcal{I}$ and $\mathcal{I}'$ iff for any $s \sim s'$ we have*

1. *$\mathcal{I}, s \models p$ iff $\mathcal{I}', s' \models p$ for all $p \in \Phi_0$.*
2. *For all $g \in \Pi_0$: If $s r_g t$, then there is a $t' \in S'$ such that $s' r'_g t'$ and $t \sim t'$.*
3. *For all $g \in \Pi_0$: If $s' r'_g t'$, then there is a $t \in S$ such that $s r_g t$ and $t \sim t'$.*

Two states are bisimilar iff there is a bisimulation $\sim$ such that $s \sim s'$. $\lhd$

One of the basic results on bisimulation in the context of modal logic states that bisimilar states make exactly the same modal formulas true. In order to extend this result to a logic like *PDL*, one shows that the three clauses in the definition of bisimulation can be generalized from atomic formulas and atomic

games/programs to arbitrary formulas and games/programs. When we say that an operation is *safe for bisimulation*, we mean exactly this, namely that clauses (2.) and (3.) can be generalized to games containing that operation.

Striving towards such a safety result for GL, we face the problem that for non-atomic games π, we have no r_π relation but only a R_π function. In order to circumvent this problem, we propose here an alternative version of bisimulation which is expressed in terms of the R_g functions. Incidentally, the following definition has been proposed in [vBvES93] as a notion of bisimulation for Concurrent Propositional Dynamic Logic.

Definition 2 (Bisimulation) *As in definition 1, but replacing the last two clauses by*

2. *For all $g \in \Pi_0$: If $s \in R_g(X)$ then $\exists X' \subseteq S'$ such that $s' \in R'_g(X')$ and $\forall x' \in X' \exists x \in X : x \sim x'$.*
3. *For all $g \in \Pi_0$: If $s' \in R'_g(X)$ then $\exists X \subseteq S$ such that $s \in R_g(X)$ and $\forall x \in X \exists x' \in X' : x \sim x'$.* ◁

It remains to show that these two definitions really amount to the same thing. Having shown this, we can turn towards the main topic of this paper, safety for bisimulation.

Theorem 1 *A relation is a bisimulation in the sense of definition 1 iff it is a bisimulation in the sense of definition 2.*

Proof. Suppose a relation $\sim$ is a bisimulation in the sense of definition 1, and assume that $s \sim s'$ and $s \in R_g(X)$, i.e. for some state t we have sr_gt and $t \in X$. Hence there is some t' such that $s'r'_gt'$ and $t \sim t'$. But then $s' \in R'_g(\{t'\})$, so $\sim$ is also a bisimulation according to definition 2.

Conversely, suppose a relation $\sim$ is a bisimulation in the sense of definition 2, and assume that $s \sim s'$ and sr_gt. Then $s \in R_g(\{t\})$ and so there is some X' such that $s' \in R'_g(X')$ and $\forall x' \in X' : t \sim x'$. Hence, there is some t' such that $s'r'_gt'$ and $t' \in X'$ and thus $t \sim t'$.

□

4 Safety & Invariance for Bisimulation

With the reformulation of bisimulation which definition 2 provides, we can now prove that in Game Logic, bisimilar states satisfy the same formulas. As mentioned before, the crucial part of the proof consists of showing that all the game constructions of GL are safe for bisimulation, i.e. that clauses (2.) and (3.) of definition 2 can be generalized to non-atomic games.

Theorem 2 *Let $\mathcal{I} = (S, \{r_a | a \in \Pi_0\}, V)$ and $\mathcal{I}' = (S', \{r'_a | a \in \Pi_0\}, V')$ be two Kripke models such that $s \in S$ and $s' \in S'$ are bisimilar. Then*

1. *For all* $\varphi \in \Phi$*:* $\mathcal{I}, s \models \varphi$ *iff* $\mathcal{I}', s' \models \varphi$
2. *For all* $\pi \in \Pi$*: If* $s \in R_\pi(X)$ *then* $\exists X' \subseteq S'$ *such that* $s' \in R'_\pi(X')$ *and* $\forall x' \in X' \, \exists x \in X : x \sim x'$.
3. *For all* $\pi \in \Pi$*: If* $s' \in R'_\pi(X)$ *then* $\exists X \subseteq S$ *such that* $s \in R_\pi(X)$ *and* $\forall x \in X \, \exists x' \in X' : x \sim x'$.

Proof. For atomic games and formulas, the claims hold by bisimilarity. For non-atomic formulas, the boolean cases are immediate and we shall only show one direction of (1.) for $\langle\pi\rangle\varphi$. If $\mathcal{I}, s \models \langle\pi\rangle\varphi$, $s \in R_\pi(\varphi^{\mathcal{I}})$ and so (by induction hypothesis (2.) for π) there is some X' such that $s' \in R'_\pi(X')$ and for all $x' \in X'$ there is some $x \in \varphi^{\mathcal{I}}$ such that $x \sim x'$. By induction hypothesis (1.) for φ, this means that $X' \subseteq \varphi^{\mathcal{I}'}$, and so by monotonicity, $s' \in R'_\pi(\varphi^{\mathcal{I}'})$, which establishes that $\mathcal{I}', s' \models \langle\pi\rangle\varphi$.

As for proving that the game constructions of *GL* are safe for bisimulation, we shall prove (2.) for non-atomic games. Consider first the case of test $\varphi?$: If $s \in R_{\varphi?}(X) = \varphi^{\mathcal{I}} \cap X$, let $X' := \{x' | \exists x \in X : x \sim x'\}$, where $\sim$ denotes the bisimulation as usual. Then $s' \in R'_{\varphi?}(X')$ by induction hypothesis (1.) for φ, and for all $x' \in X'$ there is some $x \in X$ such that $x \sim x'$, simply by definition of X'.

For union, if $s \in R_{\alpha\cup\beta}(X)$ we can assume w.l.o.g. that $s \in R_\alpha(X)$ and apply the induction hypothesis, i.e. for some X', we have $s' \in R'_\alpha(X')$ and hence also $s' \in R'_{\alpha\cup\beta}(X')$.

For composition, suppose that $s \in R_\alpha(R_\beta(X))$. Using the induction hypothesis for α, there is some Y' such that $s' \in R'_\alpha(Y')$ and for all $y' \in Y'$ there is a $u \in R_\beta(X)$ such that $u \sim y'$. Now let $X' := \{x' | \exists x \in X : x \sim x'\}$. We must show that $s' \in R'_\alpha(R'_\beta(X'))$. For this, it suffices by monotonicity to show that $Y' \subseteq R'_\beta(X')$. So suppose that $y' \in Y'$, i.e. for some $u \in R_\beta(X)$ we have $u \sim y'$. Using the induction hypothesis for β, there is some V' such that $y' \in R'_\beta(V')$ and for all $v' \in V'$ there is some $x \in X$ such that $x \sim v'$. Hence $V' \subseteq X'$ and so by monotonicity, $y' \in R'_\beta(X')$

Dual: Suppose $s \in R_{\alpha^d}(X)$, i.e. $s \notin R_\alpha(\overline{X})$. Again, let $X' := \{x' | \exists x \in X : x \sim x'\}$. It is sufficient to show that $s' \notin R'_\alpha(\overline{X'})$. Suppose by reductio the contrary. Then there is some Z with $s \in R_\alpha(Z)$ and for all $z \in Z$ there is some $x' \notin X'$ such that $z \sim x'$. From this it follows that $Z \subseteq \overline{X}$, so by monotonicity $s \in R_\alpha(\overline{X})$, a contradiction.

Iteration: Let $X' := \{x' | \exists x \in X : x \sim x'\}$ and $Z := \{z | \forall z' : \; z \sim z' \Rightarrow z' \in R'_{\alpha^*}(X')\}$. Now it is sufficient to show that $R_{\alpha^*}(X) \subseteq Z$, and given the definition of $R_{\alpha^*}(X)$ as a least fixpoint, it suffices to show that Z is a fixpoint, i.e. that $X \cup R_\alpha(Z) \subseteq Z$. Supposing that $x \in X$ and for some x' we have $x \sim x'$, we have $x' \in X' \subseteq R'_{\alpha^*}(X')$. On the other hand, suppose that $x \in R_\alpha(Z)$ and $x \sim x'$. Then by induction hypothesis, there is some Z' such that $x' \in R'_\alpha(Z')$ and for all $z' \in Z'$ there is some $z \in Z$ such that $z \sim z'$. But then $Z' \subseteq R'_{\alpha^*}(X')$, and so by monotonicity $x' \in R'_\alpha(R'_{\alpha^*}(X')) \subseteq R'_{\alpha^*}(X')$ which completes the proof.

□

References

[d'A98] Giovanna d'Agostino. *Modal Logic and Non-Well-founded Set Theory: bisimulation, translation, and interpolation*. PhD thesis, University of Amsterdam, 1998.

[vB76] Johan van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, 1976.

[vB93] Johan van Benthem. Program constructions that are safe for bisimulation. Report CSLI-93-179, CSLI, 1993.

[vB96] Johan van Benthem. Bisimulation: The never-ending story. In J. Tromp, editor, *A Dynamic and Quick Intellect. Liber Amicorum Paul Vitány*, pages 23–27. CWI, 1996.

[vB97] Johan van Benthem. Modality, bisimulation and interpolation in infinitary logic. Report ML-97-06, Institute for Logic, Language and Computation, University of Amsterdam, 1997.

[vBB93] Johan van Benthem and Jan Bergstra. Logic of transition systems. ILLC Prepublication Series CT-93-03, University of Amsterdam, 1993.

[vBvES93] J. van Benthem, J. van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. Computer Science Report CS-R9321, CWI, 1993.

[Har84] David Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II. D. Reidel Publishing Company, 1984.

[Hol98] Marco Hollenberg. *Logic and Bisimulation*. PhD thesis, University of Utrecht, 1998.

[KT90] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. MIT Press, 1990.

[Par85] Rohit Parikh. The logic of games and its applications. In Marek Karpinski and Jan van Leeuwen, editors, *Topics in the Theory of Computation*, volume 24 of *Annals of Discrete Mathematics*. Elsevier, 1985.