Bisimulations in Game Analysis

Roland Glück

Institut für Informatik, Universität Augsburg,
Universitätsstr. 6a, D-86159 Augsburg, Germany
glueck@informatik.uni-augsburg.de
http://www.informatik.uni-augsburg.de/en/chairs/dbis/pmi/staff/glueck

Abstract. Bisimulations are widely used in model checking for proving system properties. However, they are also useful for constructing systems with certain desired properties. As an example we show how strategies for a simple class of games can be synthesized with the help of bisimulations.

1 Introduction

In recent papers ([3, 4]) it was demonstrated how bisimulations can be used to construct systems with certain liveness, security and optimality properties by refining a given system. The main idea to show the correctness of this approach was that bisimulations preserve the dynamics of a system in a suited sense. The approach makes sense if the used bisimulation decreases the number of states of the given system. In [1] a method for constructing a winning strategy (if there is one) for a class of two-player games is derived. Since it relies also on the dynamics of the graph defined by the possible moves between the positions it seems promising to use bisimulations in order to reduce the size of the game graph. We will show that this idea works. First, we give an introducing example and present its solution with a short sketch of the method from [1]. Next we briefly describe bisimulations and quotients, and finally show the correctness of the resulting algorithm.

2 The Sheeps and the Wolve

2.1 Rules of the Game

A commonly known class of simple games are the so-called Fox-and-Hound games (see Google or Wikipedia for further informations). We will consider a small variant of the games described there. It is played by two players on a 3×4-chessboard as shown in the left part of Figure 1.

The white figures (symbolizing foxes or sheeps) can only move diagonally downwards on an adjacent free square, whereas the black figure (the hound or the wolve) can move one square diagonally in each direction, both upwards and downwards. The goal of the white party is to lock the black piece so that it can

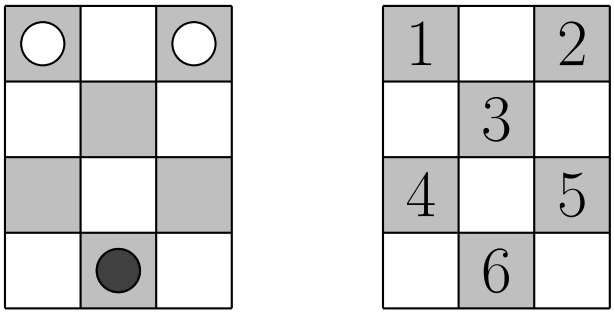

Fig. 1. Initial Position and Square Numbering

not move anymore, while the black party tries to reach the top rank of the board, i.e. one of the squares numbered with 1 or 2 in the right part of Figure 1. One player leads the white figures, one the black one. The moves are made alternately, in every move exactly one figure is moved, and each player has the obligation to make a move if possible. Here we consider the case that the white party does the first move.
To analyze the game we first construct the game graph. Its nodes are labeled with positions arising during the game. Here the label $\frac{ab}{c}$ means that the two white pieces are placed on the squares with numbers a and b, and the black piece stands on square c. Moves made by the white party are marked by lightgray arrows, those of the black one by black arrows. The game graph according to this conventions is depicted in figure 2.

Winning positions for the black side according to the above rules are marked by a surrounding double line, while winning positions for the white party wear a lightgray surrounding.
In order to give a formal defintion of a game graph we have to distinguish which moves (i.e. edges) are done by which player (from the graph above it is not clear which player has to move in a given position, although this can be determined by simple backward analysis). So as a formal definition of a game graph we choose a tupel $G = (V, (E_1, E_2))$ such that $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are two graphs over the same node set V representing the moves of player 1 or player 2 resp. If we want the players to make their move alternately we have to require that for all nodes $u, v, w \in V$ the implications $(u, v) \in E_i \wedge (v, w) \in E_j \Rightarrow i \neq j$, $(u, v) \in E_i \wedge (w, v) \in E_j \Rightarrow i = j$ and $(v, u) \in E_i \wedge (v, w) \in E_j \Rightarrow i = j$ hold. If necessary this can be easily obtained by adding a new component to each node indicating which player has to move.
A *winning strategy* in a game graph $G = (V, (E_1, E_2))$ for player 1 is a game graph $G' = (V', (E_1', E_2'))$ with $V' = V$, $E_1' \subseteq E_1$ and $E_2' = E_2$, such that every edge in E_1' starting in a winning position for player 1 ends also in a winning position for player 1 (so we prevent player 1 from making bad moves only in winning positions; we do not give him advice in losing or stalemate positions).

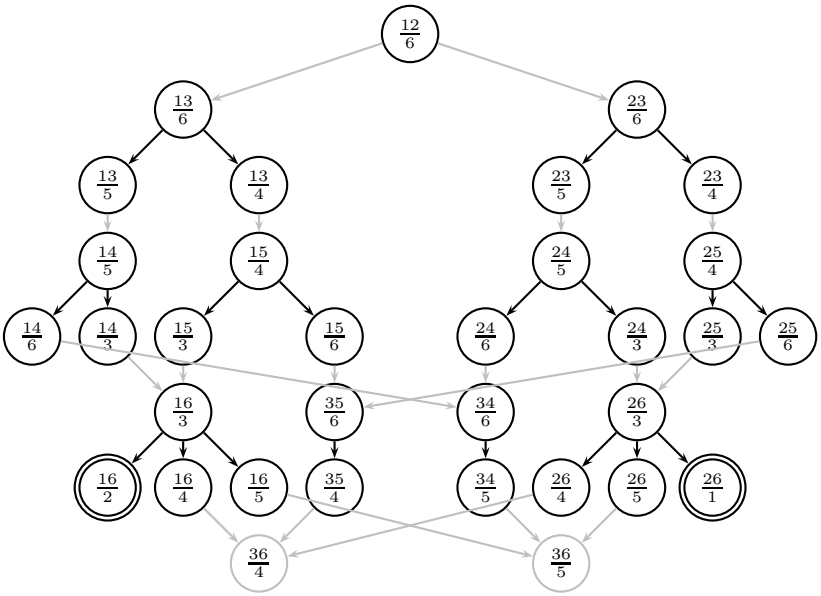

Fig. 2. The Game Graph

A winning strategy is called *maximal* if its E_1' is maximal among all winning strategies (note that the choice $E_1' = \emptyset$ delivers a winning strategy by definition).

2.2 Solution Strategy

The idea to determine winning and losing positions for each of both sides is rather simple: If it is possible for player 1 to reach a winning position from a given position the given position is also a winning position for player 1. Conversly, if every move by player 1 from a given position leads into a losing position for player 1 the given position is a losing position for player 1. We will assume that every winning position for player 1 is a losing position for player 2 and vice versa. So for example $\frac{16}{4}$ is a winning position for the white side (note that in this position also the white party is the one to move!), and $\frac{16}{2}$ is a winning position for the black side.
This idea is formalized in [1]. We will describe it shortly using the modal operators $|\cdot\rangle$ and $|\cdot]$ on graphs. For a graph $G = (V, E)$ and a subset $V' \subseteq V$ we define the *forward diamond* by $|E\rangle V' \stackrel{def}{=} \{u' \in V \mid \exists v' \in V' : (u', v') \in E\}$, i.e. $|E\rangle V'$ is the preimage of V' under E. The *forward box* is defined by $|E]V' \stackrel{def}{=} \{u' \in V \mid (u', v') \in E \Rightarrow v' \in V'\}$, with other words, $|E]V'$ is the set of all nodes from which a step along E leads by force into a node in V'. Note that a node without E-successors automatically belongs to $|E]V'$.
So, given a game graph $G = (V, (E_1, E_2))$, an inital set $W \subseteq V$ of winning

positions and an initial set $L \subseteq V$ of losing positions for player 1 (we should assume that W and L are disjoint) we can determine the winning and losing positions by the following idea: Compute repeatedly $W = |E_1\rangle W \cup |E_2]W$ and $L = |E_1]L \cup |E_2\rangle L$ until a fixpoint of (W, L) is reached (the remaining positions are stalemate positions, in which no player can enforce the win of the game). If we apply this method to our game we will see that the black party can enforce the win.
After a short glance on the game graph one will notice its symmetry. So a nearby idea will be to merge symmetric nodes into one node, afterwards to apply the above approach, and finally to unfold the obtained result. This strategy works; in the sequel we describe the necessary theoretical background for its correctness.

3 Bisimulations

3.1 Basic Definitions and Properties

Bisimulations are widely used and known (for example in model checking, see [2]); we will concentrate here on autobisimulations. An *autobisimulation* for a graph $G = (V, E)$ is a relation $B \subseteq V \times V$ with the properties $(x, y) \in E \wedge (x, x') \in B \Rightarrow \exists\, y' : (x', y') \in E \wedge (y, y') \in B$ and $(x', y') \in E \wedge (x', x) \in B^\circ \Rightarrow \exists\, y : (x, y) \in E \wedge (y', y) \in B^\circ$ for all $x, x', y, y' \in V$. Here B° is used for the converse relation of B.
Intuitively, an autobisimulation relates states with equivalent local dynamic behaviour: if there is an E-step from a state x to a state y, and x is related via B with a state x', then there has to be an E-step from x' to another state y', such that y is related with y' via B.
It is well-known that the set of autobisimulations for a fixed graph is closed under union, relational composition and converse. An autobisimulation which is also an equivalence is called *autobisimulation equivalence*.
For a graph $G = (V, E)$ and a partition $\mathcal{V} = (V_i)_{i \in I}$ of V (i.e. $V = \dot{\bigcup}_{i \in I} V_i$) we say that an autobisimulation equivalence B *respects* $\mathcal{V}$ if every set $V_i \in \mathcal{V}$ can be written as the union of suited equivalence classes of B. As above the set of autobisimulations respecting a fixed partition are closed under union, composition and converse; also the identity is an equivalence autobisimulation for every partition. So for every partition $\mathcal{V}$ there is a coarsest autobisimulation respecting $\mathcal{V}$, and it is even an autobisimulation equivalence.

3.2 Quotient and Expansion

The advantage of autobisimulation equivalences is that the number of states of a system can be decreased without violating relevant dynamic properties. This is achieved by means of quotient and expansion.
For a graph $G = (V, E)$ and an equivalence $R \subseteq V \times V$ we define the *quotient* G/R of G by R as the graph $G/R \stackrel{def}{=} (V/R, E/R)$, where V/R is the set of equivalence classes of R, and for equivalence classes V_1, V_2 of R there is an

edge $(V_1, V_2) \in E/R$ iff there is an edge $(v_1, v_2) \in E$ such that $V_1 = v_1/R$ and $V_2 = v_2/R$, where v_i/R denotes the equivalence class of v_i wrt. R. Conversely, if for a quotient $G/R = (V/R, E/R)$ a subgraph $(G/R)' = ((V/R)', (E/R)')$ with $(V/R)' = (V/R)$ and $(E/R)' \subseteq (E/R)$ is given we define the *expansion* $(G/R)'\backslash R$ as the graph $(G/R)'\backslash R \stackrel{def}{=} ((V/R)'\backslash R, (E/R)'\backslash R)$ with $(V/R)'\backslash R = V$ and $(v_1, v_2) \in (E/R)'\backslash R$ iff there is an edge $(V_1, V_2) \in (E/R)'$ with $v_1/R = V_1$ and $v_2/R = V_2$. For details see [4].
There are some relationships concerning the existence of walks in the original graph, the quotient and the expansion. Let $G = (V, E)$ be a walk, $R \subseteq V \times V$ an equivalence, and G/R, $(G/R)'$ and $(G/R)'\backslash R$ as above. First, if there is a walk $v_1 v_2 \dots v_n$ in $(G/R)'\backslash R$ then there is a walk $V_1 V_2 \dots V_n$ in $(G/R)'$ with $V_i = v_i/R$. Second, if there is a walk $V_1 V_2 \dots V_n$ in $(G/R)'$ then for all $v_1 \in V_1$ there is a walk $v_1 v_2 \dots v_n$ with $v_i \in V_i$ for $i \geq 2$. Concerning the obvious equality $(G/R)\backslash R = G$ we obtain relationships between walks in G and G/R. Also, since an edge can be seen as a walk with exactly two nodes, the above properties hold in analogous manner for edges, too.

4 Using Bisimulations in Game Analysis

4.1 Connections between Bisimulations and Modal Operators

To make use of bisimulations in our context of game analysis we will establish some connections between the modal operators $|\cdot\rangle$ and $|\cdot]$ from Subsection 2.2 and autobisimulation equivalences. If one considers the observations made at the end of Subsection 3.2 the following theorem is almost obvious:

Theorem 4.1. *Let $G = (V, E)$ be a graph, $\mathcal{V} = (V_i)_{i \in I}$ be a partition of V and B be the coarsest autobisimulation for G respecting $\mathcal{V}$. Then for every $I' \subseteq I$ the sets $|E\rangle(\bigcup_{i' \in I'} V_{i'})$ and $|E](\bigcup_{i' \in I'} V_{i'})$ can be written as the union of suited equivalence classes of R. To be exact, one has $|E\rangle(\bigcup_{i' \in I'} V_{I'}) = \bigcup_{i' \in I'} \{v \,|\, (v/R) \in |(E/R)\rangle V_{i'}\}$ and the analogous equality for the forward box.*

A short look at the solution strategy from Subsection 2.2 shows that during its execution the computed subsets can be represented as unions of equivalence classes of the coarsest autobisimulation respecting the initial partition into winning, losing and yet undefined positions. This encourages us to reduce the node number of the game graph by a bisimulation before applying our solution strategy.

4.2 Applying Bisimulations in Game Analysis

Let us return to the abstract form of our introducing example. We have a game graph $G = (V, (E_1, E_2))$ and an initial partition $\mathcal{V} = \{W, L, U\}$ of V into disjoint sets of winning (W) and losing (L) position for let us say player 1

($U = V \backslash (W \cup L)$ are positions of yet unknown character). Our goal is to determine the winning and losing positions, and to construct a maximal winning strategy. Before immediately working with bisimulations we face one difficulty: We are confronted with two graphs ((V, E_1) and (V, E_2)) instead of only one. So we have to use the coarsest relation R which is an autobisimulation both for (V, E_1) and (V, E_2) and additionally respects $\mathcal{V}$. Fortunately, this is an equivalence and enjoys the same relevant properties to make theorem 4.1 applicable. Moreover, it is efficiently computable (see [5]).
So, after computing R we first construct the two graphs $(V/R, E_1/R)$ and $(V/R, E_2/R)$. Then we solve the game on $G/R = (V/R, (E_1/R, E_2/R))$, where as winning positions in G/R we choose the set $W/R := \{w/R \,|\, w \in W\}$ and analogously the losing positions. On this graph we solve the game problem and obtain a set of winning positions $\mathcal{W} \subseteq V/R$ and losing position $\mathcal{L} \subseteq V/R$ in G/R. Furthermore, we can compute a maximal winning strategy $(G/R)' = ((V/R)', ((E_1/R)', (E_2/R)'))$ of G/R. Then the set of winning resp. losing position in G equals $\bigcup_{W \in \mathcal{W}} W$ resp. $\bigcup_{L \in \mathcal{L}} L$. The maximal winning strategy in G for player 1 is given by $(V, ((E_1/R)' \backslash R), E_2))$.

5 Conclusion and Outlook

We sketched how to use bisimulations in the context of analyzing simple two player games. The profit of their application depends on how much the state number is decreased after building the quotient.
If one is interested in similar applications the next level would be to investigate whether this approach can be used for games with more than two players. Another potential application area the various kinds of stochastic games.

References

1. R. Backhouse and D. Michaelis. Fixed-point characterisation of winning strategies in impartial games. In R. Berghammer and B. Möller, editors, *7th International Seminar on Relational Methods in Computer Science — RelMiCS 7*, volume 3051 of *Lecture Notes in Computer Science*, pages 34–47. Springer, 2004.
2. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
3. R. Glück, B. Möller, and M. Sintzoff. A semiring approach to equivalences, bisimulations and control. In R. Berghammer, A. Jaoua, and B. Möller, editors, *11th International Conference on Relational Methods in Computer Science — RelMiCS 2009*, volume 5827 of *Lecture Notes in Computer Science*, pages 134–149. Springer, 2009.
4. R. Glück, B. Möller, and M. Sintzoff. Model refinement using bisimulation quotients. In M. Johnson and D. Pavlovic, editors, *13th International Conference on Algebraic Methodology And Software Technology — AMAST 2010*, volume to be appear of *Lecture Notes in Computer Science*. Springer, 2010.
5. R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal for Computing*, 16(6).