Monotonic and Downward Closed Games*

PAROSH AZIZ ABDULLA, *Uppsala University, Sweden.*
E-mail: parosh@it.uu.se

AHMED BOUAJJANI and JULIEN D'ORSO, *University of Paris 7, France.*

Abstract

In an earlier work [Abdulla *et al.* (2000, Information and Computation, 160, 109–127)] we presented a general framework for verification of infinite-state transition systems, where the transition relation is monotonic with respect to a *well quasi-ordering* on the set of states. In this article, we investigate extending the framework from the context of transition systems to that of *games* with infinite state spaces. We show that *monotonic* games with safety winning conditions are in general undecidable. In particular, we show this negative results for games which are defined over Petri nets. We identify a subclass of monotonic games, called *downward closed* games. We provide algorithms for analysing downward closed games subject to safety winning conditions. We apply the algorithm to games played on lossy channel systems. Finally, we show that weak parity games are undecidable for the above classes of games.

Keywords: infinite-state games, lossy channel systems, Vector Addition Systems, Petri nets.

1 Introduction

One of the main challenges undertaken by the model checking community has been to develop algorithms which can deal with infinite state spaces. In a previous work [1], we presented a general framework for verification of infinite-state *transition systems*. The framework is based on the assumption that the transition relation is monotonic with respect to a *well quasi-ordering* on the set of states (configurations). The framework has been used both to give uniform explanations of existing results for infinite-state systems such as Petri nets, Timed automata [2], lossy channel systems [5], and relational automata [7, 9]; and to derive novel algorithms for model checking of Broadcast protocols [12, 13], timed Petri nets [6] and cache coherence protocols [11], etc.

A related approach to model checking is that of control [3]. Behaviours of reactive systems can naturally be described as *games* [10, 21], where control problems can be reduced to the problem of providing winning strategies. Since the state spaces of reactive systems are usually infinite, it is relevant to try to design algorithms for solving games over infinite state spaces.

We consider *turn-based* games, played between two players A and B, and investigate two types of winning conditions: (i) *safety* winning conditions in which player A tries to avoid a given set of final (bad) configurations, while player B tries to force the play into such a configuration; and (ii) *weak parity* winning conditions, where each configuration is equipped with a *rank* chosen from a finite set of natural numbers. The winning condition is defined by the parity of the lowest rank of a configuration appearing in the play. For simplicity, we sometimes refer to games with the above winning conditions as safety and weak parity games, respectively.

In this article, we investigate extending the framework of [1] from the context of transition systems to that of games with infinite state spaces. This turns out to be non-trivial. In fact, for one of the simplest classes of monotonic games, namely those induced by *Petri nets*, and for the simplest possible

*A preliminary version of this article appeared in Proc. CSL/KGC'03, Computer Science Logic and 8th Kurt Gödel Colloquium, 2003.

Vol. 18 No. 1, © The Author, 2007. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oxfordjournals.org
Published online 22 November 2007 doi:10.1093/logcom/exm062

winning condition, namely safety conditions, we show that it is undecidable to check whether a configuration is winning for a given player. On the other hand, we show decidability of safety games for a subclass of monotonic games, namely *downward closed games*: if a player can make a move from a configuration c_1 to another configuration c_2, then all configurations which are larger than c_1 (with respect to the ordering on the state space) can also make a move to c_2. Typical examples of downward closed systems are those with *lossy behaviours* such as lossy channel systems [5] and lossy VASS [8]. For these models, a configuration c_3 which is larger than c_1 can perform a silent transition to c_1 from which it can move to c_2. This behaviour explains the name *downward closed* which we use for this class of games.

We summarize our (un)decidability results as follows:

- Decidability of safety games where player B has a downward closed behaviour (a *B-downward closed game*). Considering the case where only one player is downward closed is relevant, since it allows, for instance, modeling behaviours of systems where one player (representing the environment) may lose messages in a lossy channel system (a so-called B-LCS game). In case player A has a deterministic behaviour (has no choices), our algorithm for B-downward closed games degenerates to the symbolic backward algorithm presented in [1, 5] for checking safety properties. It turns out that the sets of winning and losing configurations are regular languages provided that the set of target configurations are regular. In fact, we show that these sets are effectively constructable. Observe that the decidability result implies decidability of the special case when both players have downward closed behaviours.
- Decidability of safety games for A-downward closed games. In case player B has a deterministic behaviour, our algorithm for A-downward closed games degenerates to the forward algorithms described in [1, 5] and [14, 15] for checking eventuality properties (of the form $\forall \diamond p$). Although, the sets of winning and losing configurations are both regular, we show (in contrast to B-downward closed games), that these sets are not effectively constructible.
- Decidability of safety properties for downward closed games do not extend to monotonic games. In particular, we show that checking safety properties for games based on VASS (Vector Addition Systems with States) is undecidable. A VASS is an extended finite-state automaton which operates on a finite set of weak counters (and are hence computationally equivalent to the classical model of Petri nets). The undecidability result holds even if both players are assumed to have monotonic behaviours.
- Undecidability of weak parity games for both A- and B-downward closed games. In particular, we show undecidability of weak parity games for both A-LCS and B-LCS games. On the other hand, if both players can lose messages, the problem is decidable.

In the first two cases, the winning strategy for the relevant player is positional.

It is worth noting that Raskin *et al.* [20] have shown decidability for another class of monotonic games, namely those based on systems modeled with a counter abstraction. This class is different from the class of downward closed games which we consider in this article.

Outline. In the next Section, we recall some basic definitions for games. In Section 3, we introduce monotonic and downward closed games. We present a symbolic algorithm for solving *B-downward closed* games with safety winning conditions in Section 4; and apply the algorithm to B-LCS in Section 5. In Section 6, we consider *A-downward closed* games. In Section 7, we show that safety monotonic games are undecidable. In Section 8, we study decidability of weak parity games for the above models. Finally, we give some conclusions and remarks in Section 9.

2 Preliminaries

In this section, we recall some standard definitions for games. We consider *turn-based* games, played between two players A and B.

A game G is a tuple $\left(C, C_A, C_B, \longrightarrow, C_F\right)$, where C is a (possibly infinite) set of *configurations*, C_A, C_B is a partitioning of C, $\longrightarrow \subseteq \left(C_A \times C_B\right) \cup \left(C_B \times C_A\right)$ is a set of *transitions*, and $C_F \subseteq C_A$ is a set of *final configurations*. The sets C_A and C_B represent the configurations of A and B. From a configuration in C_A, Player A is allowed to move to a configuration in C_B. The choices of player A should be consistent with the transition relation. Player B makes moves in an analogous manner.

We write $c_1 \longrightarrow c_2$ to denote that $\left(c_1, c_2\right) \in \longrightarrow$. For a configuration c, we define $Pre(c) = \left\{c' \mid c' \longrightarrow c\right\}$, and define $Post(c) = \left\{c' \mid c \longrightarrow c'\right\}$. We extend *Pre* to sets of configurations such that $Pre(D) = \cup_{c \in D} Pre(c)$. The function *Post* can be extended in a similar manner. Without loss of generality, we assume that there is no deadlock, i.e. $Post(c) \neq \emptyset$ for each configuration c. For a set $D \subseteq C_A$ of configurations, we define $\overset{A}{\neg} D$ to be the set $C_A \setminus D$. The operator $\overset{B}{\neg}$ is defined in a similar manner. For a set $D \subseteq C_A$, we use $\widetilde{Pre}(D)$ to denote $\overset{B}{\neg}\left(Pre\left(\overset{A}{\neg} D\right)\right)$. For $E \subseteq C_B$, we define $\widetilde{Pre}(E)$ in a similar manner.

A *play* P (of G) from a configuration c is an infinite sequence $c_0, c_1, c_2, \ldots$ of configurations such that $c_0 = c$, and $c_i \longrightarrow c_{i+1}$, for each $i \geq 0$. We will consider *safety games* in which player A tries to avoid states in C_F (such a game can also be seen as a *reachability games* in which player B tries to force the play into C_F). A play $c_0, c_1, c_2, \ldots$ is *winning* for player A if there is no $j \geq 0$ with $c_j \in C_F$.

A *strategy* for player A (or simply an A-*strategy*) is a partial function $\sigma_A : C_A \mapsto C_B$ such that $c \longrightarrow \sigma_A(c)$. A B-*strategy* is a partial function $\sigma_B : C_B \mapsto C_A$ and is defined in a similar manner to an A-strategy. A configuration $c \in C_A$ together with strategies σ_A and σ_B (for players A and B respectively) define a play $P(c, \sigma_A, \sigma_B) = c_0, c_1, c_2, \ldots$ from c where $c_0 = c$, $c_{2i+1} = \sigma_A(c_{2i})$, and $c_{2i+2} = \sigma_B(c_{2i+1})$, for $i \geq 0$. A similar definition is used in case $c \in C_B$ (interchanging the order of applications of σ_A and σ_B to the configurations in the sequence).

An A-strategy σ_A is said to be *winning* for player A from a configuration c, if for all B-strategies σ_B, it is the case that $P(c, \sigma_A, \sigma_B)$ is winning for player A. A configuration c is said to be *winning* for player A if there is a winning A-strategy from c. We shall consider the *safety problem* for games:

The safety problem

Instance A game G with safety winning conditions and a configuration c.

Question Is c winning for player A?

Observe that solving the safety game for player A can be done by solving the reachability game for player B. In the latter, player B tries to force the game into a configuration belonging C_F. It is well known that safety/reachability games are positionally determined (see e.g. [16]).

3 Ordered games

In this section, we introduce *monotonic* and *downward closed* games.

Orderings Let A be a set and let $\preceq$ be a quasi-order (i.e. a reflexive and transitive binary relation) on A. We say that $\preceq$ is a *well quasi-ordering (wqo)* on A if there is no infinite sequence $a_0, a_1, a_2, \ldots$ with $a_i \not\preceq a_j$ for $i < j$. Consider a well quasi-ordering $\preceq$. For $B \subseteq A$, we say that B is *canonical* if there are no $a, b \in B$ with $a \neq b$ and $a \preceq b$. We use *min* to denote a function where, for $B \subseteq A$, the value of $min(B)$ is a canonical subset of B such that for each $b \in B$ there is $a \in min(B)$ with $a \preceq b$. We say that

$\preceq$ is *decidable* if, given $a,b \in A$ we can check whether $a \preceq b$. A set $B \subseteq A$ is said to be *upward closed* if $a \in B$ and $a \preceq b$ imply $b \in B$. A *downward closed* set is defined in an analogous manner.

Monotonic games An *ordered game* G is a tuple $\left(C, C_A, C_B, \longrightarrow, C_F, \preceq\right)$, where $\left(C, C_A, C_B, \longrightarrow, C_F\right)$ is a game and $\preceq \subseteq \left(C_A \times C_A\right) \cup \left(C_B \times C_B\right)$ is a decidable wqo on the sets C_A and C_B. The ordered game G is said to be *monotonic* with respect to player A (or simply A*-monotonic*) if, for each $c_1, c_2 \in C_A$ and $c_3 \in C_B$, whenever $c_1 \preceq c_2$ and $c_1 \longrightarrow c_3$, there is a c_4 with $c_3 \preceq c_4$ and $c_2 \longrightarrow c_4$. A B*-monotonic game* is defined in a similar manner. A *monotonic game* is both A-monotonic and B-monotonic.

Downward closed games An ordered game $G = \left(C, C_A, C_B, \longrightarrow, C_F, \preceq\right)$ is said to be A*-downward closed* if, for each $c_1, c_2 \in C_A$ and $c_3 \in C_B$, whenever $c_1 \longrightarrow c_3$ and $c_1 \preceq c_2$, then $c_2 \longrightarrow c_3$. A B*-downward closed game* is defined in a similar manner. A game is *downward closed* if it is both A- and B-downward closed. Notice that each class of downward closed games is included in the corresponding class of monotonic games. For instance, each A-downward closed game is A-monotonic. From the definitions we get the following property.

LEMMA 3.1
For an A-downward closed game G and any set $E \subseteq C_B$, the set $Pre(E)$ is upward closed. An analogous result holds for B-downward closed games.

4 B-Downward closed games

In this section, we present a standard scheme for solving the safety problem [22]. We instantiate the scheme to obtain a symbolic algorithm for solving B-downward closed safety games. In the rest of this section, we assume a B-downward closed game $G = \left(C, C_A, C_B, \longrightarrow, C_F, \preceq\right)$.

Scheme Given a configuration c in G, we want to decide whether c is winning for player A or not. To do that, we introduce a scheme by considering a sequence s of sets of configurations of the form:

$$s: \quad D_0 \,,\, E_0 \,,\, D_1 \,,\, E_1 \,,\, D_2 \,,\, E_2 \,, \ldots$$

where $D_i \subseteq C_A$ and $E_i \subseteq C_B$. Intuitively, the sets D_i and E_i characterize the configurations (in C_A and C_B respectively) which are winning for player B. The elements of the sequence are defined by

$$D_0 = C_F \qquad E_0 = Pre(D_0)$$

$$D_{i+1} = D_i \cup \widetilde{Pre}(E_i) \quad E_{i+1} = E_i \cup Pre(D_{i+1}) \quad i = 0,1,2,\ldots$$

We say that *s converges (at ℓ)* if $D_{\ell+1} \subseteq D_\ell$ or $E_{\ell+1} \subseteq E_\ell$. In such a case, the set $D_\ell \cup E_\ell$ characterizes exactly the set of configurations which are winning for player B. The question of whether a given configuration c is winning for player A amounts therefore to whether $c \notin \left(D_\ell \cup E_\ell\right)$. The sets D_i and E_i are referred to in [16] as the *attractors* of the set C_F. To show that our characterization is correct, we show the following two lemmas. The first lemma shows that if c appears in one of the generated sets then it is a winning configuration for player B. The second lemma states that if the sequence converges, then the generated sets contain all configurations which are winning for player B.

LEMMA 4.1
If $c \in D_i \cup E_i$, for some $i \geq 0$, then c is winning for player B.

PROOF. For a play $P = c_0, c_1, c_2, \ldots$ and a configuration c with $c \longrightarrow c_0$, we use $c \bullet P$ to denote the play $c, c_0, c_1, c_2, \ldots$.

We let $D'_0 = D_0$, $E'_0 = E_0$, $D'_{i+1} = D_{i+1} \setminus D_i$, and let $E'_{i+1} = E_{i+1} \setminus E_i$.

We define a B-strategy σ_B such that, for each $c \in \cup_{i \geq 0} D_i \cup E_i$ and A-strategy σ_A, the play $P(c, \sigma_A, \sigma_B)$ is winning for player B. We can take σ_B to be any B-strategy such that, for each $i \geq 0$ and $c \in E'_i$, we have $\sigma_B(c) = c'$ for some $c' \in D_i$. By definition of E'_i, we know that such a c' exists, and that $\sigma_B(c)$ is well-defined.

Let σ_A be any A-strategy and let c be any configuration with $c \in D_i \cup E_i$. We show that $P\big(c, \sigma_A, \sigma_B\big)$ is winning for player B. We use induction on the positions of the sets D_i and E_i in the sequence s above.

Base Case If $c \in D_0$ then the result follows from the definitions.

Induction Step We show that $P(c, \sigma_A, \sigma_B)$ is winning for player B. We consider two cases. First, we show the case where $c \in E_i$. If $i > 0$ and $c \in E_{i-1}$ then it follows by the induction hypothesis that $P(c, \sigma_A, \sigma_B)$ is winning for player B. Otherwise, we know that $c \in E'_i$, which means that $\sigma_B(c) = c'$ for some $c' \in D_i$. By the induction hypothesis we know that $P(c', \sigma_A, \sigma_B)$ is winning for player B. It follows that $P(c, \sigma_A, \sigma_B) = c \bullet P(c', \sigma_A, \sigma_B)$ is winning for player B.

Next, we consider the case where $c \in D_{i+1}$. If $c \in D_i$ then it follows by the induction hypothesis that $P(c, \sigma_A, \sigma_B)$ is winning for player B. Otherwise, let $\sigma_A(c) = c'$. Since $c \in D_{i+1}$, we know that $c' \in E_i$. By the induction hypothesis it follows that $P(c', \sigma_A, \sigma_B)$ is winning for player B. This means that $P(c, \sigma_A, \sigma_B) = c \bullet P(c', \sigma_A, \sigma_B)$ is winning for player B. ■

LEMMA 4.2
If s converges at ℓ and $c \notin D_\ell \cup E_\ell$ then c is winning for player A.

PROOF. Suppose that s converges at ℓ. We define an A-strategy σ_A such that, for each $c \notin \cup_{i \geq 0} \big(D_i \cup E_i\big)$ and B-strategy σ_B, the play $P(c, \sigma_A, \sigma_B)$ is winning for player A. We can take σ_A to be any A-strategy such that, for each $c \in \overset{A}{\neg} \big(\cup_{i \geq 0} D_i\big)$, we have $\sigma_A(c) = c'$ for some $c' \in \overset{B}{\neg} \big(\cup_{i \geq 0} E_i\big)$. Such a c' exists according to the following argument: Suppose that c' does not exist. This means that, for each $c' \in Post(c)$, there is a j such that $c' \in E_j$. Since s converges at ℓ, it follows that $Post(c) \subseteq E_\ell$. This implies that $c \in D_{\ell+1}$ which is a contradiction.

Let σ_B be any B-strategy and let c be any configuration with $c \notin \cup_{i \geq 0} \big(D_i \cup E_i\big)$. We show that $P\big(c, \sigma_A, \sigma_B\big) = c_0, c_1, c_2, \ldots$ is winning for player A. First, we show the following property: for each $j \geq 0$ we have $c_j \notin \cup_{i \geq 0} \big(D_i \cup E_i\big)$. We use induction on j.

Base Case Trivial.

Induction Step We consider two cases. If $c_{j+1} \in C_A$. Suppose that $c_{j+1} \in D_i$ for some $i : 0 \leq i \leq \ell$. This means that $c_j \in E_i$ which contradicts the induction hypothesis.

If $c_{j+1} \in C_B$. Suppose that $c_{j+1} \in E_i$ for some $i \geq 0$. We know that $c_{j+1} = \sigma_A(c_j)$. From the definition of σ_A it follows that $c_j \in D_k$ for some $k : 0 \leq k \leq \ell$, which again is a contradiction to the induction hypothesis.

Now we can prove the lemma. Suppose that $P\big(c, \sigma_A, \sigma_B\big) = c_0, c_1, c_2, \ldots$ is winning for player B. This means that there is a $j \geq 0$ such that $c_j \in C_F$. This implies that $c_j \in D_0$ which is a contradiction to the above property. ■

Below, we present a symbolic algorithm based on the scheme above. We shall work with *constraints* which we use as symbolic representations of sets of configurations.

Constraints An *A-constraint* denotes a (potentially infinite) set $[\![\phi]\!] \subseteq C_A$ of configurations. A *B-constraint* is defined in a similar manner. For constraints ϕ_1 and ϕ_2, we use $\phi_1 \sqsubseteq \phi_2$ to denote that $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$. For a set Φ of constraints, we use $[\![\Phi]\!]$ to denote $\bigcup_{\phi \in \Phi} [\![\phi]\!]$. For sets of constrains Φ_1 and Φ_2, we use $\Phi_1 \sqsubseteq \Phi_2$ to denote that $[\![\Phi_2]\!] \subseteq [\![\Phi_1]\!]$. Sometimes, we identify constraints with their interpretations, so we write $c \in \phi$, $\phi_1 \subseteq \phi_2$, $\phi_1 \cap \phi_2$, $\neg\phi$, etc. We consider a particular class of

B-constraints which we call *upward closed constraints*. A constraint ϕ is said to be *upward closed* if $[\![\phi]\!]$ is upward closed with respect to $\preceq$, i.e., $c \in [\![\phi]\!]$ and $c \preceq c'$ implies $c' \in [\![\phi]\!]$.

A set Ψ of constraints is said to be *effective* with respect to the game G if

- The set C_F is characterized by a finite set $\Phi_F \subseteq \Psi$, i.e., $[\![\Phi_F]\!] = C_F$.
- For a configuration c and a constraint $\phi \in \Psi$, we can decide whether $c \in [\![\phi]\!]$. For finite sets of constraints Φ_1, Φ_2, we can decide whether $\Phi_1 \sqsubseteq \Phi_2$.
- For each A-constraint $\phi \in \Psi$, we can compute a finite set Φ' of upward closed B-constraints such that $[\![\Phi']\!] = Pre\big([\![\phi]\!]\big)$. In such a case we use $Pre(\phi)$ to denote the set Φ'. Notice that $Pre\big([\![\phi]\!]\big)$ is upward closed by Lemma 3.1. Also, observe that computability of $Pre(\phi)$ implies that, for a finite set $\Phi \subseteq \Psi$, we can compute a finite set Φ' of upward closed constraints such that $[\![\Phi']\!] = Pre\big([\![\Phi]\!]\big)$. We use $Pre(\Phi)$ to denote the set Φ'.
- For each finite set Φ of B-constraints, we can compute a finite set $\Phi' \subseteq \Psi$ of A-constraints such that $[\![\Phi']\!] = \widetilde{Pre}\big([\![\Phi]\!]\big)$. In such a case we use $\widetilde{Pre}(\Phi)$ to denote the set Φ'.

The game G is said to be *effective* if there is a set Ψ of constraints which is effective with respect to G.

Symbolic algorithm Given a constraint system Ψ which is effective with respect to the game G, we can solve the safety game problem by deriving a symbolic algorithm from the scheme described earlier. Each D_i will be characterized by a finite set of A-constraints $\Phi_i \subseteq \Psi$, and each E_i will be represented by a finite set of B-constraints Φ'_i. More precisely:

$$\begin{array}{lll} \Phi_0 = \Phi_F & \Phi'_0 = Pre(\Phi_0) & \\ \Phi_{i+1} = \Phi_i \cup \widetilde{Pre}(\Phi'_i) & \Phi'_{i+1} = \Phi'_i \cup Pre(\Phi_{i+1}) & i=0,1,2,\ldots \end{array}$$

The algorithm terminates in case $\Phi'_j \sqsubseteq \Phi'_{j+1}$. In such a case, a configuration c is winning for player B if and only if $c \in [\![\Phi_j]\!] \cup [\![\Phi'_j]\!]$. This gives an effective procedure for deciding the safety game problem according to the following

- Since Ψ is effective with respect to G we can:
  - perform each step of the algorithm.
  - check whether $c \in [\![\phi]\!]$, for any configuration c and constraint ϕ in Φ_i or Φ'_i.
  - check the termination condition.
- Termination is guaranteed due to well quasi-ordering of $\preceq$ as follows. Suppose that the algorithm does not terminate. Then, there is an infinite sequence $\Phi'_0, \Phi'_1, \Phi'_2, \ldots$, where each Φ'_i is a finite set of B-constraint, such that $\Phi'_i \not\sqsubseteq \Phi'_{i+1}$ for each $i \geq 0$. This means that $[\![\Phi'_j]\!] \not\subseteq [\![\Phi'_{i+1}]\!]$ for each $i \geq 0$. By definition, we know that $[\![\Phi'_0]\!] \subseteq [\![\Phi'_1]\!] \subseteq [\![\Phi'_2]\!] \subseteq \cdots$. It follows that $[\![\Phi'_0]\!] \subset [\![\Phi'_1]\!] \subset [\![\Phi'_2]\!] \subset \cdots$. In other words the are configurations $c_0, c_1, c_2, \ldots$ where $c_j \in [\![\Phi'_j]\!]$ and $c_j \notin [\![\Phi'_i]\!]$ if $j < i$. Since Φ'_i is a set of B-constraints, it follows that $[\![\Phi'_i]\!]$ is upward closed for each $i \geq 0$. This implies that $c_i \not\preceq c_j$ if $j > i$, and hence the sequence of configurations violates the assumption that the set C_B is well quasi-ordered.

From this we get the following:

THEOREM 4.3
The safety problem is decidable for the class of effective B-downward closed games in case the set C_F is characterized by a finite set of constraints.

5 B-Lossy channel systems

In this section, we apply the symbolic algorithm presented in Section 4 to solve the safety game problem for *B-LCS* games: games between two players operating on a finite set of channels (unbounded FIFO buffers), where player B is allowed to lose any number of messages before each move.

For a function f, we use $f[\ell := a]$ to denote the function f' such that $f'(\ell) = a$ and $f'(\ell') = f(\ell')$ if $\ell' \neq \ell$.

For a finite set M and words $x_1, x_2 \in M^*$, we use $x_1 \bullet x_2$ to denote the concatenation of x_1 and x_2. For $x \in M^*$ and $X \subseteq M^*$, we use $X \bullet x$ to denote the set of words $x' \bullet x$ with $x' \in X$. Also, we use $x^{-1} \bullet X$ to denote the left quotient of X with respect to x, i.e. to denote the set of words x' such that $x \bullet x' \in X$. For $x_1, x_2 \in M^*$, we use $x_1 \preceq x_2$ to denote that x_1 is a (not necessarily contiguous) substring of x_2.

A *B-lossy channel system (B-LCS)* is a tuple $(S, S_A, S_B, L, M, T, S_F)$, where S is a finite set of *(control) states*, S_A, S_B is a partitioning of S, L is a finite set of *channels*, M is a finite *message alphabet*, T is a finite set of *transitions*, and $S_F \subseteq S_A$ is the set of *final states*. Each transition in T is a triple (s_1, op, s_2), where

- either $s_1 \in S_A$ and $s_2 \in S_B$, or $s_1 \in S_B$ and $s_2 \in S_A$.
- *op* is of one of the forms: $\ell!m$ (sending message m to channel ℓ), or $\ell?m$ (receiving message m from channel ℓ), or *nop* (not affecting the contents of the channels).

A B-LCS $\mathcal{L} = (S, S_A, S_B, L, M, T, S_F)$ induces a B-downward closed game $G = (C, C_A, C_B, \longrightarrow, C_F, \preceq)$ as follows:

- *Configurations:* Each configuration $c \in C$ is a pair (s, w), where $s \in S$, and w, called a *channel state*, is a mapping from L to M^*. In other words, a configuration is defined by the control state and the contents of the channels. We partition the set C into $C_A = \left\{ (s, w) \mid s \in S_A \wedge w \in L^{M^*} \right\}$ and $C_B = \left\{ (s, w) \mid s \in S_B \wedge w \in L^{M^*} \right\}$.
- *Final configurations:* The set C_F is defined to be $\{(s, w) \mid s \in S_F\}$.
- *Ordering:* For channel states w_1, w_2, we use $w_1 \preceq w_2$ to denote that $w_1(\ell) \preceq w_2(\ell)$ for each $\ell \in L$. For configurations $c_1 = (s_1, w_1)$ and $c_2 = (s_2, w_2)$, we use $c_1 \preceq c_2$ to denote that both $s_1 = s_2$ and $w_1 \preceq w_2$.
  The ordering $\preceq$ is decidable and wqo (by Higman's Lemma [17]).
  For a set D of configurations, we use $D\uparrow$ to denote the upward closure of D with respect to $\preceq$. In other words, $D\uparrow$ is the set $\{c' \mid \exists c \in D.\, c \preceq c'\}$.
- *Non-loss transitions*: $(s_1, w_1) \longrightarrow (s_2, w_2)$ if one of the following conditions is satisfied
  - There is a transition in $t \in T$ of the form $(s_1, \ell!m, s_2)$, and w_2 is the result of appending m to the head of $w_1(\ell)$, i.e., $w_2 = w_1[\ell := m \bullet w_1(\ell)]$.
  - There is a transition in $t \in T$ of the form $(s_1, \ell?m, s_2)$, and w_2 is the result of removing m from the end of $w_1(\ell)$, i.e., $w_1 = w_2[\ell := w_1(\ell) \bullet m]$.
  - There is a transition in $t \in T$ of the form (s_1, nop, s_2), and $w_2 = w_1$.

  In the above three cases we use $t(s_1, w_1)$ to denote (s_2, w_2) and use $t^{-1}(s_2, w_2)$ to denote (s_1, w_1). For a set C of configurations and a transition $t \in T$, we use $t(C)$ to denote the set $\{c_2 \mid \exists c_1 \in C.\, t(c_1) = c_2\}$. We define $t^{-1}(C)$ in a similar manner.
- *Loss transitions:* If $s_1 \in S_B$ and $(s_1, w_1) \longrightarrow (s_2, w_2)$ according to one of the previous three rules then $(s_1', w_1') \longrightarrow (s_2, w_2)$ for each (s_1', w_1') with $(s_1, w_1) \preceq (s_1', w_1')$.

Remark. To satisfy the condition that there are no deadlock states in games induced by B-LCS, we will add a number of states. The idea is that the player who deadlocks is considered to be a loser. More precisely, we add two 'winning' states $s_1^* \in S_A$ $s_2^* \in S_B$ for player A. We also add two 'winning' states for player B, namely $s_3^* \in S_A$, $s_4^* \in S_B$, where $s_3^* \in S_F$, and $s_1^* \notin S_F$. We add four transitions $\left(s_1^*, nop, s_2^*\right)$, $\left(s_2^*, nop, s_1^*\right)$, $\left(s_3^*, nop, s_4^*\right)$, and $\left(s_4^*, nop, s_3^*\right)$. Furthermore, we add transitions $\left(s, nop, s_4^*\right)$ for each $s \in S_A$, and $\left(s, nop, s_1^*\right)$ for each $s \in S_B$. Intuitively, if player A enters a configuration, where he has no other options, then he is forced to move to s_4^* losing the game. A similar reasoning holds for player B.

We show decidability of the safety problem for B-LCS using Theorem 4.3. In order to do that we introduce *regular constraints* which are effective with respect to B-LCS. A *regular constraint* over M is a finite-state automaton (or equivalently a regular expression) characterizing a regular set over M. A regular constraint ϕ over channel states is a mapping from L to regular constraints over M, with an interpretation $[\![\phi]\!] = \{w \mid \forall \ell \in L.\, w(\ell) \in [\![\phi(\ell)]\!]\}$. A regular constraint ϕ (over configurations) is of the form $\left(s, \phi'\right)$, where $s \in S$ and ϕ' is a regular constraint over channel states, with an interpretation $[\![\phi]\!] = \left\{\left(s, w\right) \mid w \in [\![\phi']\!]\right\}$.

Next we show that regular constraints are effective for B-LCS games (Lemma 5.4). First, we show three auxiliary lemmas.

Lemma 5.1
For a regular constraint $\left(s_2, \phi_2\right)$ and a transition t, we can effectively compute a regular constraint which denotes $t^{-1}(s_2, \phi_2)$.

Proof. If t is of the form $\left(s_1, \ell!m, s_2\right)$ then $t^{-1}(s_2, \phi_2) = \left(s_1, \phi_2[\ell := m^{-1} \bullet \phi_2(\ell)]\right)$. The result follows from the fact that regular languages are closed under taking the left quotient with respect to a word (and in particular with respect to a symbol of the alphabet).

If t is of the form $\left(s_1, \ell?m, s_2\right)$ then $t^{-1}(s_2, \phi_2) = \left(s_1, \phi_2[\ell := \phi_2(\ell) \bullet m]\right)$. The result follows since regular languages are closed under concatenation.

If t is of the form $\left(s_1, nop, s_2\right)$ then $t^{-1}(s_2, \phi_2) = \left(s_1, \phi_2\right)$. The result follows from regularity of ϕ_2. ■

Lemma 5.2
For a regular constraint ϕ_1, we can effectively compute a regular constraint ϕ_2 such that $\phi_2 = \phi_1 \uparrow$.

Proof. For a regular language L_1 represented by a finite-state automaton A_1, we can construct a new automaton A_2 representing $L_1 \uparrow$. We can derive A_2 from A_1 by adding a self-loop for each symbol in the alphabet and each state in A_1. ■

Lemma 5.3
For a regular constraint ϕ_1, we can effectively compute a regular constraint ϕ_2 such that $\phi_2 = \neg\phi_1$.

Proof. For a regular constraint ϕ over channel states (with a set L of channels and a set M of messages), the complement $\neg\phi$ is the set of constraints ϕ' where there is an $\ell \in L$ such that $\phi'(\ell) = \neg\phi(\ell)$ and $\phi'(\ell') = M^*$ if $\ell' \neq \ell$. For a regular constraint $\phi_1 = \left(s, \phi\right)$, the complement $\neg\phi_1$ is given by the set of constraints of the form $\phi_2 = \left(s', \phi'\right)$ where either $s' \neq s$ or $\phi' \in \neg\phi$. ■

Lemma 5.4
Regular constraints are effective for B-LCS games.

Proof.
- The set C_F is characterized by the (finite) set of constraints of the form $\left(s, \phi\right)$ where $s \in S_F$ and $\phi(\ell) = M^*$ for each $\ell \in L$. This set is obviously regular.

- For a configuration c and a regular constraint ϕ we can check whether $c \in [\![\phi]\!]$ as it amounts to checking membership of a word in a regular set. For regular constraints $\phi_1, \phi_2 \in \Psi$, we can check whether $\phi_1 \sqsubseteq \phi_2$ as it amounts to deciding inclusion between regular languages.
- For each regular A-constraint ϕ, we can compute a finite set Φ of upward closed B-constraints, such that $\Phi = Pre(\phi)$. The result follows from Lemma 5.1, Lemma 5.2 and the fact that $Pre(\phi) = \left(\bigcup_{t \in T} t^{-1}(\phi)\right)\uparrow$.
- For each finite set Φ_1 of regular B-constraints, we can compute a finite set Φ_2 of regular A-constraints, such that $\Phi_2 = \widetilde{Pre}(\Phi_1)$. We recall that $\widetilde{Pre}(\Phi) = \overset{A}{\neg}\left(Pre\left(\overset{B}{\neg}\Phi\right)\right)$. The result follows from Lemma 5.1 and Lemma 5.3.

■

From Theorem 4.3 and Lemma 5.4 we get the following:

THEOREM 5.5
The safety problem is decidable for B-LCS games.

6 *A*-Downward closed games

We present an algorithm for solving the safety problem for A-downward closed games. We use the algorithm to prove decidability of the safety problem for a variant of lossy channel games, namely *A-LCS*.

An A-downward closed game is said to be *executable* if for each configuration c, the set $Post(c)$ is finite and computable. Observe that this implies that the game is finitely branching.

Suppose that we want to check whether a configuration $c_{\text{init}} \in C_A$ is winning for player A. The algorithm builds an AND-OR tree, where each node of the tree is labelled with a configuration. OR-nodes are labelled with configurations in C_A, while AND-nodes are labelled with configurations in C_B.

We build the tree successively, starting from the root, which is labelled with c_{init} (the root is therefore an OR-node). At each step we pick a leaf with label c and perform one of the following operations:

- If $c \in C_F$ then we declare the node *unsuccessful* and close the node (we will not expand the tree further from the node).
- If $c \in C_B$, $c \notin C_F$, and there is predecessor node in the tree with label c' where $c' \preceq c$ then we declare the node *successful* and close the node.
- Otherwise, we add a set of successors, each labelled with an element in $Post(c)$. This step is possible by the assumption that the game is executable.

The procedure terminates by Köning's Lemma and by well quasi-ordering of $\preceq$. The resulting tree is evaluated interpreting AND-nodes as conjunction, OR-nodes as disjunction, successful leaves as the constant *true* and unsuccessful leaves as the constant *false*. More precisely, we compute a function *eval* which takes as input a node n in the tree and returns a Boolean value as follows:

- If n is successful then $eval(n) = true$.
- If n is unsuccessful then $eval(n) = false$.
- Otherwise, let $\{n_1, \ldots, n_m\}$ be the set of children of n, and let c be the label of n. If $c \in C_A$ then let $eval(n) := eval(n_1) \vee \cdots \vee eval(n_m)$; and if $c \in C_A$ then let $eval(n) := eval(n_1) \wedge \cdots \wedge eval(n_m)$.

Notice that, when the tree is completely built, a node is a leaf iff it has been declared successful/unsuccessful. The algorithm answers 'yes' if and only if $eval(n_{\text{root}}) = true$ where n_{root} is the root of the tree. We show that the construction is correct. More precisely, depending on whether $eval(n_{\text{root}})$ is *true* resp. *false*, we construct a winning strategy for player A resp. B.

Suppose that $eval(n_{\text{root}}) = true$. We define a winning strategy σ_A for player A. We let σ_A to be any strategy which satisfies the following properties. For each $c \in C_A$ such that there is an interior node n of the tree with $eval(n) = true$ and label c, we consider a node n' which is child of n and with $eval(n') = true$ (such an n' exists by definition). We define $\sigma_A(c) := c'$, where c' is the label of n'.

Suppose that $eval(n_{\text{root}}) = false$. We define a winning strategy σ_B for player B. For each $c \in C_B$ which is the label of an interior node we define $\sigma_B(c)$ in a similar manner to above (replacing the constant *true* by *false*). For a configuration c which is the label of an unsuccessful node n, let n' be a predecessor node of n such that n' is labeled with a configuration $c' \preceq c$ (such an n' exists by definition). We let $\sigma_B(c) := \sigma_B(c')$.

From this, we get the following.

THEOREM 6.1
The safety problem is decidable for executable A-downward closed games.

A-LCS An *A-LCS* has the same syntax as a B-LCS. The difference is that it is now player A who loses messages (rather than player B). The game induced by an A-LCS has a similar behaviour to that induced by a B-LCS. The difference is that in the definition of the *loss transitions*:

- If $s_1 \in S_A$ and $(s_1, w_1) \longrightarrow (s_2, w_2)$ according to a non-loss transition then $(s'_1, w'_1) \longrightarrow (s_2, w_2)$ for each (s'_1, w'_1) with $(s_1, w_1) \preceq (s'_1, w'_1)$.

It is straightforward to check that a game induced by an A-LCS is A-downward closed and executable. This gives the following.

THEOREM 6.2
The safety problem is decidable for A-LCS games.

Although the safety problem is decidable for A-LCS games, it is not possible to compute a characterization of the set of winning configurations as we did for B-LCS. This is shown as follows. Observe that the set of winning configurations for player A is upward closed. Therefore, this set can be characterized by a finite set of regular constraints. However, we show that we cannot compute a finite set of regular constraints Φ such that $[\![\Phi]\!]$ is the set of winning configurations for player A. To do that, we use a result reported in [8] for transition systems induced by lossy channel systems. The result in [8] implies that we cannot compute a finite set of regular constraints characterizing the set of configurations c satisfying the property $c \models \exists_\infty \Box \neg S_F$, i.e. we cannot compute a finite set of regular constraints characterizing the set of configurations from which there is an infinite computation which never visits a given set S_F of control states. Given a lossy channel system $\mathcal{L}$ (inducing a transition system) and a set S_F of states, we derive an A-LCS $\mathcal{L}'$ (inducing an A-downward closed game). For each configuration c in $\mathcal{L}$, it is the case that $c \models \exists_\infty \Box \neg S_F$ if and only if the configuration corresponding to c is winning in the game induced by $\mathcal{L}'$. Intuitively, player A simulates the transitions of $\mathcal{L}$, while player B follows passively. More precisely, each state s in $\mathcal{L}$ has a copy $s \in C_A$ in $\mathcal{L}'$. For each transition $t = (s_1, op, s_2)$ in $\mathcal{L}$, there is a corresponding 'intermediate state' $s_t \in C_B$ and two corresponding transitions (s_1, op, s_t) and (s_t, nop, s_2) in $\mathcal{L}'$. Furthermore, we have two state $s_1^* \in C_A$ and $s_2^* \in C_B$ which are losing for player A (defined in a similar manner to Section 5). Each configuration in C_A can perform a transition labelled with *nop* to s_2^*. It is straightforward to check that a configuration c is winning for player A in $\mathcal{L}'$ if and only if $c \models \exists_\infty \Box \neg S_F$.

Notice that this implies we cannot either compute a finite set of regular constraints characterizing the set of winning configurations for player B.

From this, we get the following:

THEOREM 6.3
We cannot compute finite sets of regular constraints characterizing the sets of winning configurations for players A and B in an A-LCS (although such sets always exist).

7 Undecidability of monotonic games

We show that the decidability of the safety problem does not extend from downward closed games to monotonic games. We show undecidability of the problem for a particular class of monotonic games, namely *VASS games*. The proof is inspired by ideas from Jančar showing undecidability of bisimulation for Petri nets [18]. In the definition of VASS games below, both players are assumed to have monotonic behaviours. Obviously, this implies undecidability for A- and B-monotonic games.

In fact, it is sufficient to consider VASS with two dimensions (two variables). Let $\mathcal{N}$ and $\mathcal{I}$ denote the set of natural numbers and integers respectively.

VASS Games A *(2-dimensional) VASS (Vector Addition System with States) game* $\mathcal{V}$ is a tuple (S, S_A, S_B, T, S_F), where S is a finite set of *(control) states*, S_A, S_B is a partitioning of S, T is a finite set of *transitions*, and $S_F \subseteq S$ is the set of *final states*. Each transition is a triple $(s_1, (a,b), s_2)$, where

- either $s_1 \in S_A$ and $s_2 \in S_B$, or $s_1 \in S_B$ and $s_2 \in S_A$.
- $a, b \in \mathcal{I}$. The pair (a,b) represents the change made to values of the variables during the transition.

A VASS $\mathcal{V} = (S, S_A, S_B, T, S_F)$ induces a monotonic game $G = (C, C_A, C_B, \longrightarrow, C_F, \preceq)$ as follows:

- Each configuration $c \in C$ is a triple (s,x,y), where $s \in S$ and $x, y \in \mathcal{N}$. In other words, a configuration is defined by the state and the values assigned to the variables.
- $C_A = \{(s,x,y) \mid (s \in S_A) \wedge (x \geq 0) \wedge (y \geq 0)\}$.
- $C_B = \{(s,x,y) \mid (s \in S_B) \wedge (x \geq 0) \wedge (y \geq 0)\}$.
- $(s_1, x_1, y_1) \longrightarrow (s_2, x_2, y_2)$ iff $(s_1, (a,b), s_2) \in T$, $x_2 = x_1 + a$, and $y_2 = y_1 + b$. Observe that since $x_2, y_2 \in \mathcal{N}$, we implicitly require $x_2 \geq 0$ and $y_2 \geq 0$; otherwise the transition is blocked.
- $C_F = \{(s,x,y) \mid (s \in S_F) \wedge (x \geq 0) \wedge (y \geq 0)\}$.
- $(s_1, x_1, y_1) \preceq (s_2, x_2, y_2)$ iff $s_1 = s_2$, $x_1 \leq x_2$, and $y_1 \leq y_2$.

We can avoid deadlock in VASS games in a similar manner to Section 5.

THEOREM 7.1
The safety problem is undecidable for VASS games.

Undecidability is shown through a reduction from an undecidable problem for *2-counter machines*.

2-Counter machines A *2-counter machine* M is a tuple (S_M, T_M), where S_M is a finite set of *states*, and T_M is a finite set of *transitions*. Each transition is a triple of the form $(s_1, (a,b), s_2)$, or $(s_1, x = 0?, s_2)$, or $(s_1, y = 0?, s_2)$, where $s_1, s_2 \in S_M$.

A *configuration* of M is a triple (s,x,y) where $s \in S_M$ and $x, y \in \mathcal{N}$. We define a transition relation $\longrightarrow$ on configurations such that $(s_1, x_1, y_1) \longrightarrow (s_2, x_2, y_2)$ iff either

- $(s_1, (a,b), s_2) \in T_M$, and $x_2 = x_1 + a$, $y_2 = y_1 + b$, $x_2 \geq 0$, and $y_2 \geq 0$; or
- $(s_1, x = 0?, s_2) \in T_M$, $x_2 = x_1 = 0$, and $y_2 = y_1$; or
- $(s_1, y = 0?, s_2) \in T_M$ and $x_2 = x_1$, and $y_2 = y_1 = 0$.

The *2-counter reachability problem* is defined as follows

2-Counter reachability problem

Instance A 2-counter machine $M=\left(S_M,T_M\right)$ and two states $s_{\text{init}},s_f\in S_M$.

Question Is there a sequence

$\left(s_0,x_0,y_0\right)\longrightarrow\left(s_1,x_1,y_1\right)\longrightarrow\left(s_2,x_2,y_2\right)\longrightarrow\cdots\longrightarrow\left(s_n,x_n,y_n\right)$

of transitions such that $s_0=s_{\text{init}}$, $x_0=0$, $y_0=0$, and $s_n=s_f$?

It is well known that the 2-counter reachability problem is undecidable. In the following, we show how to reduce the 2-counter reachability problem to the safety problem for VASS games. Given a 2-counter machine $M=\left(S_M,T_M\right)$ and two states $s_{\text{init}},s_f\in S_M$, we construct a corresponding VASS game $\mathcal{V}$, such that the reachability problem over M has a positive answer if and only if the safety game is winning over $\mathcal{V}$ for player B. Intuitively, player B emulates the moves of M, while player A is passive. Tests for equality with 0 cannot be emulated directly by a VASS system. This means that player B could try to make moves not corresponding to an actual move of the 2-counter machine. However, if player B tries to 'cheat', i.e. to make a forbidden move, then we allow player A to go into a winning escape loop. This means that player B always chooses to make legal moves. Furthermore, we add an escape loop accessible when the system has reached the final state. This loop is winning for player B. Thus, player B wins whenever the final state is reachable. Formally, we define the VASS game $\mathcal{V}=\left(S,S_A,S_B,T,S_F\right)$ as follows:

- $S_A=\left\{s_t^A\,|\,t\in T_M\right\}\cup\left\{s_*^A,s_{\text{reached}}^A,s_{\text{init}}^A\right\}$. In other words, for each transition $t\in T_M$ there is a state $s_t^A\in S_A$. We also add three special states s_*^A,s_{reached}^A and s_{init}^A to S_A.
- $S_B=\left\{s^B\,|\,s\in S_M\right\}\cup\left\{s_*^B\right\}$. In other words, for each state in $s\in S_M$ there is a corresponding state $s^B\in S_B$. We also add a special state s_*^B to S_B.
- For each transition t of the form $\left(s_1,\left(a,b\right),s_2\right)\in T_M$, there are two transitions in T, namely $\left(s_1^B,\left(a,b\right),s_t^A\right)$ and $\left(s_t^A,\left(0,0\right),s_2^B\right)$. Player B chooses a move, and player A follows passively.
- For each transition t of the form $\left(s_1,x=0?,s_2\right)\in T_M$, there are three transitions in T, namely $\left(s_1^B,\left(0,0\right),s_t^A\right)$, $\left(s_t^A,\left(0,0\right),s_2^B\right)$, and $\left(s_t^A,\left(-1,0\right),s_*^B\right)$. Player B may cheat here. However, if this is the case, player A will be allowed to move to s_*^B, which is winning.
- Transitions of the form $\left(s_1,y=0?,s_2\right)\in T_M$ are handled in a similar manner to the previous case.
- There are five additional transitions in T, namely an initializing transition $\left(s_{\text{init}}^A,\left(0,0\right),s_{\text{init}}^B\right)$; an escape loop to detect that the final state has been reached $\left(s_f^B,\left(0,0\right),s_{\text{reached}}^A\right)$ and $\left(s_{\text{reached}}^A,\left(0,0\right),s_f^B\right)$; a loop to detect illegal moves $\left(s_*^B,\left(0,0\right),s_*^A\right)$ and $\left(s_*^A,\left(0,0\right),s_*^B\right)$.
- $S_F=\left\{s_{\text{reached}}^A\right\}$.

Let $G=\left(C,C_A,C_B,\longrightarrow,C_F,\preceq\right)$ be the monotonic game induced by $\mathcal{V}$. We show that there is a sequence

$\left(s_0,x_0,y_0\right)\longrightarrow\left(s_1,x_1,y_1\right)\longrightarrow\left(s_2,x_2,y_2\right)\longrightarrow\cdots\longrightarrow\left(s_n,x_n,y_n\right)$ of transitions in M with $s_0=s_{\text{init}}$, $x_0=0$, $y_0=0$, and $s_n=s_f$ iff the configuration $\left(s_{\text{init}}^A,0,0\right)$ is winning for player B in G subject to safety winning conditions.

(if) Suppose that $\left(s_{\text{init}}^A,0,0\right)$ is winning in the safety game over G for player B. Then for any A-strategy, there exists a B-strategy such that $P\left(\left(s_{\text{init}}^B,0,0\right),\sigma_A,\sigma_B\right)$ is winning for player B. This is true in particular if we choose the following A-strategy. We define $\sigma_A(\left(s_t^A,x,y\right))=c$, where c is:

- if $t=\left(s_1,\left(a,b\right),s_2\right)$ then $c=\left(s_2^B,x,y\right)$.
- if $t=\left(s_1,x=0?,s_2\right)$ and $x>0$ then $c=\left(s_*^B,x-1,y\right)$.

- if $t=(s_1,x=0?,s_2)$ and $x=0$ then $c=(s_2^B,x,y)$.
- the cases for $t=(s_1,y=0?,s_2)$ are defined in a similar manner as the two previous cases.

Consider $P((s_{\text{init}}^B,0,0),\sigma_A,\sigma_B)=c_0,c_1,c_2,\ldots$. Since $P((s_{\text{init}}^B,0,0),\sigma_A,\sigma_B)$ is winning for player B, there is a $k:0\leq k\leq length(P)$ such that c_k is of the form $(s_{\text{reached}}^A,x,y)$. Such a configuration can only be reached if the predecessor c_{k-1} is of form $\left(s_f^B,x,y\right)$. By the construction of $\mathcal{V}$ it follows that there is no $j\leq k$ where c_j is of the form (s_*^A,x,y) or (s_*^B,x,y) (otherwise the definition of σ_A would imply that $P((s_{\text{init}}^A,0,0),\sigma_A,\sigma_B)$ is winning for player A). This means that $P((s_{\text{init}}^A,0,0),\sigma_A,\sigma_B)$ is in fact of the form

$(s_{\text{init}}^A,x_0,y_0),(s_0^B,x_0,y_0),(s_{t_1}^A,x_1,y_1),(s_2^B,x_2,y_2),(s_{t_3}^A,x_3,y_3),\ldots$

By the construction of $\mathcal{V}$ we know that

$(s_0,x_0,y_0)\longrightarrow(s_2,x_2,y_2)\longrightarrow\cdots$

is a sequence of transitions of our 2-counter machine. Since $s_{k-1}=s_f$, we know that we have found the desired sequence.

(only if) Suppose that there is a sequence of transitions of the above form. For each $i:0\leq i<n$ we know that there is a transition $t_i=(s_i,(a_i,b_i),s_{i+1})$. Also, we can assume without loss of generality that $(s_i,x_i,y_i)\neq(s_j,x_j,y_j)$ if $i\neq j$ in the sequence above.

We now define a B-strategy σ_B such that $P((s_{\text{init}}^B,0,0),\sigma_A,\sigma_B)$ is winning for player B against any A-strategy σ_A. We define σ_B to be any B-strategy such that $\sigma_B((s_i^B,x_i,y_i))=(s_{t_i}^A,x_i+a_i,y_i+b_i))=(s_{t_i}^A,x_{i+1},y_{i+1}))$, for each $i:0\leq i<n$. We observe that for any A-strategy we have $\sigma_A((s_{t_i}^A,x_{i+1},y_{i+1}))=\left(s_{i+1}^B,x_{i+1},y_{i+1}\right)$. This follows from the fact that transitions to s_*^B are never enabled during the above sequence.

8 Weak parity games

A *weak parity game G of degree n* is a tuple $(C,C_A,C_B,\longrightarrow,r)$ where $C,C_A,C_B,\longrightarrow$ are defined as in games (Section 2), and r is a mapping from C to the set $\{0,\ldots,n\}$ of natural numbers. We use C^k to denote $\{c|\,r(c)=k\}$. The sets C_A^k and C_B^k are defined in a similar manner. We call $r(c)$ the *rank* of c. Abusing notation, we define the *rank* $r(P)$ of a play $P=c_0,c_1,c_2,\ldots$ to be $\min\{r(c_0),r(c_1),r(c_2)\ldots\}$. We say that P is *weak parity winning* for player A if $r(P)$ is even. We say that c is *weak parity winning* for player A if there is an A-strategy σ_A such that, for each B-strategy σ_B, it is the case that $P(c,\sigma_A,\sigma_B)$ is weak parity winning for player A.

The weak parity problem

Instance A weak parity game G and a configuration c in G.

Question Is c (weak parity) winning for player A?

We show below that the weak parity problem is undecidable for A-downward closed games. In particular, we show undecidability of the problem for A-LCS games. The proof for B-downward closed games is similar.

THEOREM 8.1
The weak parity problem is undecidable for A-LCS games.

In [4] we show undecidability of the *infinite computation problem*, for transition systems based on lossy channel systems.

The infinite computation problem

Instance A lossy channel systems $\mathcal{L}$ and a control states s_{init}.

Question Is there a channel state w such that there is an infinite computation starting from (s_{init}, w)?

We reduce the infinite computation problem for LCS to the weak parity problem for A-LCS. We construct a new $\mathcal{L}'$ to simulate $\mathcal{L}$. Intuitively, we let player A choose the moves of the original system, while player B follows passively. An additional loop at the beginning of $\mathcal{L}'$ allows us to guess the initial contents w of the channels. If the system deadlocks, then player B wins. So the only way for player A to win is to make the system follow an infinite sequence of moves. More formally, $\mathcal{L}' = (S, S_A, S_B, L, M, T, S_F)$ is defined as follows. For each control state s in $\mathcal{L}$, we create a control state $s^A \in S_A$. For each transition t in $\mathcal{L}$, we create a control state $s_t^B \in S_B$. For each transition $t = (s_1, op, s_2)$ in $\mathcal{L}$ there are two transitions (s_1^A, op, s_t^B) and (s_t^B, nop, s_2^A) in $\mathcal{L}'$. Furthermore, there are five additional states $s_1^*, s_4^* \in S_A$, $s_2^*, s_3^*, s_5^* \in S_B$, together with the following transitions:

- Two transitions $(s_1^*, \ell!m, s_2^*)$ and (s_2^*, nop, s_1^*) for each $m \in M$ and $\ell \in L$. These two allow to build up the initial channel contents.
- Two transitions (s_1^*, nop, s_3^*) and $(s_3^*, nop, s_{\text{init}}^A)$. This is to get to the initial state of $\mathcal{L}$ when the channel content is ready.
- A transition (s^A, nop, s_5^*) for each control state s in $\mathcal{L}$. This transition is only taken when $\mathcal{L}$ is deadlocked.
- Two transitions (s_4^*, nop, s_5^*), and (s_5^*, nop, s_4^*). This loop indicates a deadlock in $\mathcal{L}$.

The ranks of the configurations are defined as follows:

- $r((s_1^*, w)) = r((s_2^*, w)) = r((s_3^*, w)) = 3$, for each w.
- $r((s^A, w)) = r((s_t^B, w)) = 2$, for each w, each transition t in $\mathcal{L}$, and each control state s in $\mathcal{L}$.
- $r((s_4^*, w)) = r((s_5^*, w)) = 1$, for each w.

We show that (s_1^*, ϵ) is weak parity-winning for player A if and only if there exists a w and an infinite sequence starting from (s_{init}, w).

(if) Suppose that there exists an infinite sequence of configurations of $\mathcal{L}$ $(s_0, w_0), (s_1, w_1), (s_2, w_2), \ldots$ with a corresponding sequence of transitions $t_i = (s_i, op_i, s_{i+1})$.

If there are repetitions in this sequence, we pick $i < j$ with $(s_i, w_i) = (s_j, w_j)$ such that there is no repetition in the sequence $(s_0, w_0), \ldots, (s_{j-1}, w_{j-1})$ We complete the sequence with repetitions of the loop $(s_i, w_i), \ldots, (s_j, w_j)$.

The sequence above has the property that either all configurations visited are unique, or that whenever a configuration is repeated, it always has the same successor.

We choose a sequence of send operations

$$l_1!m_1, l_2!m_2, \ldots, l_n!m_n$$

such that from empty channels, we get to w_0. Let w^i be the channel contents after applying send operations 1 through i.

Then we define an A-strategy σ_A to be any strategy satisfying the following:

- $\sigma_A((s_1^*, w^i)) = (s_2^*, w^{i+1})$, i.e. stay in the rank 3 loop to build the initial channel contents w_0;
- $\sigma_A((s_1^*, w_0)) = (s_3^*, w_0)$, i.e., when the channel content is complete, exit the loop, and go to s_{init};
- $\sigma_A((s_i^A, w_i)) = (s_{t_i}^B, w_{i+1})$, i.e., follow the sequence of transitions of $\mathcal{L}$.

Observe that σ_A is well-defined, since each configuration visited has a uniquely defined successor.

For any B-strategy, we can see that the play $P((s_1^*, \epsilon), \sigma_A, \sigma_B)$ stays indefinitely long in the rank 2 zone. Thus, the rank of the entire play is 2, and the initial configuration is weak parity-winning.

(only if) Let us assume that (s_1^*,ϵ) is weak parity-winning for the A-strategy σ_A. By construction of $\mathcal{L}'$, we know that the rank of a play $P((s_1^*,\epsilon),\sigma_A,\sigma_B)$, for any B-strategy σ_B, has to be 2. This means that our play never reaches a configuration with s_4^* or s_5^* (otherwise, the play would have rank 1, and wouldn't be winning). Since the rank of any configuration with s_1^* or s_2^* is 3, we deduce that our play can be split into two parts:

- a finite part with rank 3 $(s_1^*,\epsilon),\ldots,(s_3^*,w_0)$;
- and an infinite part of rank 2 $(s_3^*,w_0),(s_{\text{init}}^A,w_0),(s_1,w_1),\ldots$.

By construction of $\mathcal{L}'$, we deduce that the sequence of configurations

$$(s_{\text{init}},w_0),(s_1,w_1),\ldots$$

is a valid sequence of transitions of $\mathcal{L}$, and it is infinite.
Remark. The above construction implies also undecidability in the case of monotonic games with Büchi and co-Büchi conditions. In the case of Büchi winning conditions we can take the set of final states to be those with rank 2. In the case of co-Büchi winning conditions we can take the set of final states to be those with rank 1 or 3.

9 Conclusions and remarks

We have considered a class of games played over infinite state spaces (infinite graphs), where the movements of the players are monotonic with respect to a given well quasi-ordering on the set of states. We show that the decidability results reported in [1] for well quasi-ordered transition systems do not carry over to the context of games. In fact, undecidability holds for VASS games (which are monotonic), even under safety winning conditions. On the other hand, we show decidability of safety properties for the class of games where the moves of one player are downward closed with respect to the ordering on the state space.

We also consider weak parity games where the rank of a play is decided by the configurations which appear *at least once* in the play. This is a simpler condition than that of the standard definition which considers configurations appearing *infinitely often*. We show undecidability for LCS games under weak parity conditions (implying undecidability also in the case of the standard definition). Using results in [19], we can strengthen this undecidability result and extend it to *lossy counter* games. Such games are special cases of LCS games where the message alphabet is of size one (each channel behaves as a *lossy counter*). However, in case both players can lose messages, we can show that the parity problem is decidable. The reason is that the best strategy for each player is to empty the channels after the next move. The problem can therefore be reduced into an equivalent problem over finite-state graphs. Notice that the undecidability proof for parity games uses a rank set of size 3 (configurations have ranks 1, 2 or 3). If the rank set is restricted to have size 2 or 3 the problem degenerates to the safety problem for B-LCS and A-LCS, respectively, giving decidability in both cases.

References

[1] P. A. Abdulla, K. Čerāns, B. Jonsson, and Y.-K. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, **160**, 109–127, 2000.

[2] R. Alur and D. Dill. Automata for modelling real-time systems. In *Proceedings of ICALP '90*. Vol. 443 of *Lecture Notes in Computer Science*, pp. 322–335. Springer, Warwick, 1990.

[3] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science*, pp. 100–109. IEEE Computer Society, Miami Beach, Florida, 1997.

[4] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, **130**, 71–90, 1996.

[5] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, **127**, 91–101, 1996.

[6] P. A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proc. ICATPN'2001: 22nd International Conference on Application and Theory of Petri nets*. Vol. 2075 of *Lecture Notes in Computer Science*, pp. 53–70. Springer, Newcastle, UK, 2001.

[7] J. M. Barzdin, J. J. Bicevskis, and A. A. Kalninsh. Automatic construction of complete sample systems for program testing. In *IFIP Congress, 1977*, Toronto, North-Holland, 1977.

[8] A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *Symposium on Theoretical Aspects of Computer Science*. Vol. 1563 of *Lecture Notes in Computer Science*, pp. 323–333. Springer, Trier, Germany, 1999.

[9] K. Čerāns. Deciding properties of integral relational automata. In *Proc. ICALP '94, 21st International Colloquium on Automata, Languages, and Programming*, Serge Abiteboul and Eli Shamir, eds. Vol. 820 of *Lecture Notes in Computer Science*, pp. 35–46. Springer Verlag, Jerusalem, 1994.

[10] L. de Alfaro, T. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Proc. CONCUR 2001, 12th International Conference on Concurrency Theory*. K. G. Larsen and M. Nielsen, eds. Vol. 2154 of *Lecture Notes in Computer Science*, pp. 536–550. Springer Verlag, Aaalborg, 2001.

[11] G. Delzanno. Automatic verification of cache coherence protocols. In *Proceedings 12th International Conference on Computer Aided Verification*, E. Allen Emerson and A. Prasad Sistla, eds. Vol. 1855 of *Lecture Notes in Computer Science*, pp. 53–68. Springer Verlag, Chicago, 2000.

[12] G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In *Proc. CSL'99*. Springer, Madrid, 1999.

[13] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. LICS '99, 14th IEEE International Symposium on Logic in Computer Science*, IEEE Computer Society, Trento, Italy, 1999.

[14] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, **7**, 129–135, 1994.

[15] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, **256**, 63–92, 2001.

[16] E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Vol. 2500 of *Lecture Notes in Computer Science*, Springer, Dagstuhl, Germany, 2002.

[17] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of London Mathematical Society*, **2**, 326–336, 1952.

[18] P. Jančar. Undecidability of bisimilarity for Petri nets and related problem. *Theoretical Computer Science*, **148**, 281–301, 1995.

[19] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*. Number 1776 in *Lecture Notes in Computer Science*, Springer, Punta del Este, Uruguay, 2000.

[20] J.-F. Raskin, M. Samuelides, and L. Van Begin. Games for counting abstractions. *Electronic Notes in Theoretical Computer Science*, **128**, 69–85, 2005.

[21] W. Thomas. Infinite games and verification. In *Proceedings of 14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pp. 58–64. Springer, Copenhagen, Denmark, 2002.

[22] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, **200**, 135–183, 1998.

Received 19 September 2005