Monotonic and Downward Closed Games[*]

Parosh Aziz Abdulla
Uppsala University, Sweden

Ahmed Bouajjani
University of Paris 7, France

Julien d'Orso
University of Paris 7, France

Abstract

In an earlier work [AČJYK00] 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 paper, we investigate extending the framework from the context of transition systems to that of *games* with infinite state spaces. We show that *monotonic* games are in general undecidable. We identify a subclass of monotonic games, called *downward closed* games. We provide algorithms for analyzing downward closed games subject to winning conditions which are formulated as safety properties.

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 [AČJYK00] 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 [AD90], lossy channel systems [AJ96b], and relational automata [BBK77, Čer94]; and to derive novel algorithms for model checking of Broadcast protocols [EFM99, DEP99], timed Petri nets [AN01], and cache coherence protocols [Del00], etc.

A related approach to model checking is that of control [AHK97]. Behaviours of reactive systems can naturally be described as *games* [LdAR01, Tho02], where

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

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.

In this paper, we consider extending the framework of [AČJYK00] 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 transition systems, namely *Petri nets*, we show that the game problem is undecidable. The negative result holds for games with the simplest possible winning condition, namely that of *safety*. Such a game is played between two players A and B, where player A tries to avoid a given set of *bad* configurations, while player B tries to force the play into such a configuration. On the other hand, we show decidability of the safety game problem 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 [AJ96b] and lossy VASS [BM99].

We summarize our (un)decidability results as follows:

- Decidability of the safety problem for 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, modelling 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 [AJ96b, AČJYK00] for checking safety properties. In fact, we compute a characterization of the set of winning (and losing) configurations in such a game. Observe that this result implies decidability of the case when both players have downward closed behaviours.

- Decidability of the safety problem 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 [AJ96b, AČJYK00] and [Fin94, FS01] for checking eventuality properties (of the form $\forall \diamond p$). However, in contrast to B-downward closed games, we show it is not possible to compute a characterization of the set of winning (or losing) configurations.

- Decidability results for downward closed games do not extend to monotonic games. In particular we show that deciding safety properties for games based on VASS (Vector Addition Systems with States) is undecidable. The undecidability result holds even if both players are assumed to have monotonic behaviours.

- Undecidability of *parity games* for both A- and B-downward closed games. In a parity game, 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. In particular, we show undecidability of parity games for both A-LCS and B-LCS games. On the other hand, if both players can lose messages, the problem is decidable.

It is worth noting that Raskin et al [RSB04] have recently shown decidability for another class of monotonic games, namely those based on systems modelled with a counter abstraction. This class is different from the class of downward closed games which we consider in this paper.

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 the safety problem for *B-downward closed* games 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 the safety problem is undecidable for monotonic games. In Section 8, we study decidability of 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.

A game G is a tuple $(C, C_A, C_B, \longrightarrow, C_F)$, where C is a (possibly infinite) set of *configurations*, C_A, C_B is a partitioning of C, $\longrightarrow \subseteq (C_A \times C_B) \cup (C_B \times C_A)$ is a set of *transitions*, and $C_F \subseteq C_A$ is a finite set of *final configurations*. We write $c_1 \longrightarrow c_2$ to denote that $(c_1, c_2) \in \longrightarrow$. For a configuration c, we define $Pre(c) = \{c' |\, c' \longrightarrow c\}$, and define $Post(c) = \{c' |\, c \longrightarrow c'\}$. 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 $i \geq 0$. A play $c_0, c_1, c_2, \ldots$ is *winning* (for 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 σ_A. 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 $c_{2i+1} = \sigma_A(c_{2i})$, 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* from a configuration c, if for all B-strategies σ_B, it is the case that $P(c, \sigma_A, \sigma_B)$ is winning. A configuration c is said to be *winning* if there is a winning A-strategy from c. We shall consider the *safety problem* for games:

The safety problem

Instance A game G and a configuration c.

Question Is c winning?

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 $i < j$. 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 $\mathit{min}(B)$ is a canonical subset of B such that for each $b \in B$ there is $a \in \mathit{min}(B)$ with $a \preceq b$. We say that $\preceq$ is *decidable* if, given $a, b \in A$ we can decide 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 a similar manner.

Monotonic Games An *ordered game* G is a tuple $(C, C_A, C_B, \longrightarrow, C_F, \preceq)$, where $(C, C_A, C_B, \longrightarrow, C_F)$ is a game and $\preceq \subseteq (C_A \times C_A) \cup (C_B \times C_B)$ 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 $G = (C, C_A, C_B, \longrightarrow, C_F, \preceq)$ 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 $\mathit{Pre}(E)$ is upward closed. A similar result holds for B-downward closed games.

4 B-Downward Closed Games

We present a symbolic algorithm for solving the safety problem for B-downward closed games. In the rest of this Section, we assume a B-downward closed game $G = (C, C_A, C_B, \longrightarrow, C_F, \preceq)$.

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

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

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 not winning. The elements of the sequence are defined by

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

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

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 not winning. The question of whether a given configuration c is winning amounts therefore to whether $c \notin (D_\ell \cup E_\ell)$. 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 not a winning configuration. The second Lemma states that if the sequence converges, then the generated sets contain all non-winning configurations.

Lemma 4.1 If $c \in D_i \cup E_i$, for some $i \geq 0$, then c is not winning.

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 $c \in D_i \cup E_i$ and A-strategy σ_A, the play $P(c, \sigma_A, \sigma_B)$ is not winning. 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 $P\,(c, \sigma_A, \sigma_B)$ is not winning. 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 not winning. 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 not winning. 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 not winning. It follows that $P(c, \sigma_A, \sigma_B) = c \bullet P(c', \sigma_A, \sigma_B)$ is not winning.

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 not winning. 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 not winning. This means that $P(c, \sigma_A, \sigma_B) = c \bullet P(c', \sigma_A, \sigma_B)$ is not winning. □

Lemma 4.2 If s converges and $c \notin D_i \cup E_i$ for each $i \geq 0$, then c is winning.

Proof Suppose that s converges at ℓ. We define an A-strategy σ_A such that, for each $c \notin \cup_{i \geq 0} (D_i \cup E_i)$ and B-strategy σ_B, the play $P(c, \sigma_A, \sigma_B)$ is winning. We can take σ_A to be any A-strategy such that, for each $c \overset{A}{\in^{\neg}} (\cup_{i \geq 0} D_i)$, we have $\sigma_A(c) = c'$ for some $c' \overset{B}{\in^{\neg}} (\cup_{i \geq 0} E_i)$. 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 $c \notin \cup_{i \geq 0} (D_i \cup E_i)$. We show that $P(c, \sigma_A, \sigma_B) = c_0, c_1, c_2, \ldots$ is winning. First, we show the following property: for each $j \geq 0$ we have $c_j \notin \cup_{i \geq 0} (D_i \cup E_i)$. 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 $i \geq 0$. 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 $k \geq 0$, which again is a contradiction to the induction hypothesis.

Now we can prove the lemma. Suppose that $P(c, \sigma_A, \sigma_B) = c_0, c_1, c_2, \ldots$ is not winning. 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 ψ of constraints, we use $[\![\psi]\!]$ to denote $\bigcup_{\phi \in \psi} [\![\phi]\!]$. For sets of constrains ψ_1 and ψ_2, we use $\psi_1 \sqsubseteq \psi_2$ to denote that for each $\phi_2 \in \psi_2$ there is a $\phi_1 \in \psi_1$ with $\phi_1 \sqsubseteq \phi_2$. Notice that

$\psi_1 \sqsubseteq \psi_2$ implies $[\![\psi_2]\!] \subseteq [\![\psi_1]\!]$. 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 $\psi_F \subseteq \Psi$, i.e., $[\![\psi_F]\!] = C_F$.
- For a configuration c and a constraint $\phi \in \Psi$, we can decide whether $c \in [\![\phi]\!]$. For constraints $\phi_1, \phi_2 \in \Psi$, we can decide whether $\phi_1 \sqsubseteq \phi_2$.
- For each A-constraint $\phi \in \Psi$, we can compute a finite set ψ' of upward closed B-constraints such that $[\![\psi']\!] = \mathit{Pre}\,([\![\phi]\!])$. In such a case we use $\mathit{Pre}(\phi)$ to denote the set ψ'. Notice that $\mathit{Pre}\,([\![\phi]\!])$ is upward closed by Lemma 3.1. Also, observe that computability of $\mathit{Pre}(\phi)$ implies that, for a finite set $\psi \subseteq \Psi$, we can compute a finite set ψ' of upward closed constraints such that $[\![\psi']\!] = \mathit{Pre}\,([\![\psi]\!])$.
- For each finite set ψ of B-constraints, we can compute a finite set $\psi' \subseteq \Psi$ of A-constraints such that $[\![\psi']\!] = \widetilde{\mathit{Pre}}\,([\![\psi]\!])$. In such a case we use $\widetilde{\mathit{Pre}}(\psi)$ to denote the set ψ'.

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 above. Each D_i will be characterized by a finite set of A-constraints $\psi_i \in \Psi$, and each E_i will be represented by a finite set of B-constraints ψ'_i. More precisely:

$$\psi_0 = \psi_F \qquad \psi'_0 = \mathit{Pre}(\psi_0)$$

$$\psi_{i+1} = \psi_i \cup \widetilde{\mathit{Pre}}(\psi'_i) \qquad \psi'_{i+1} = \psi'_i \cup \mathit{Pre}(\psi_{i+1}) \qquad i = 0, 1, 2, \ldots$$

The algorithm terminates in case $\psi'_j \sqsubseteq \psi'_{j+1}$. In such a case, a configuration c is not winning if and only if $c \in [\![\psi_j]\!] \cup [\![\psi'_j]\!]$. 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 ψ_i or ψ'_i.
  - check the termination condition.

- Termination is guaranteed due to well quasi-ordering of $\preceq$ (which implies well quasi-ordering of $\sqsubseteq$ on upward closed constraints).

From this we get the following

Theorem 4.3 The safety problem is decidable for the class of effective B-downward closed games.

5 B-LCS

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 $C_A = \{(s, w) \,|\, s \in S_A\}$ and $C_B = \{(s, w) \mid s \in S_B\}$.

- *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 [Hig52]).

  For set C of configurations, we use $C\!\uparrow$ to denote the upward closure of C with respect to $\preceq$. In other words, $C\!\uparrow$ is the set $\{c' \mid \exists c \in C.\ c \preceq c'\}$.
- *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_1 is the result of removing m from the end of $w_2(\ell)$, i.e., $w_1 = w_2[\ell := w_2(\ell) \bullet m]$.
  - 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 can always add two "winning" states $s_1^* \in S_A$, $s_2^* \in S_B$, and two "losing" states $s_3^* \in S_A$, $s_4^* \in S_B$, where $s_3^* \in S_F$, and $s_1^* \notin S_F$. We add four transitions (s_1^*, nop, s_2^*), (s_2^*, nop, s_1^*), (s_3^*, nop, s_4^*), and (s_4^*, nop, s_3^*). Furthermore, we add transitions (s, nop, s_4^*) for each $s \in S_A$, and (s, nop, s_1^*) 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 (s, ϕ'), where $s \in S$ and ϕ' is a regular constraint over channel states, with an interpretation $[\![\phi]\!] = \{(s, w) \mid w \in [\![\phi']\!]\}$.

Next we show that regular constraints are effective for B-LCS games (Lemma 5.3). First, we show two auxiliary lemmas.

Lemma 5.1 For a regular constraint (s_2, ϕ_2) and a transition t, there is a regular constraint which denotes $t^{-1}(s_2, \phi_2)$.

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

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

Lemma 5.2 For a regular constraint ϕ_1, we can effectively a 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 Regular constraints are effective for B-LCS games.

Proof

- The set C_F is characterized by the (finite) set of constraints of the form (s, M^*) where $s \in S_F$ which 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 ψ of upward closed B-constraints, such that $\psi = \mathit{Pre}(\phi)$. The result follows from Lemma 5.1, Lemma 5.2 and the fact that $\mathit{Pre}(\phi) = \big(\bigcup_{t \in T} t^{-1}(\phi)\big)\!\uparrow$.
- For each finite set ψ_1 of regular B-constraints, we can compute a finite set ψ_2 of regular A-constraints, such that $\psi_2 = \widetilde{\mathit{Pre}}(\psi_1)$. We recall that $\widetilde{\mathit{Pre}}(\psi) = \overset{A}{\neg}\left(\mathit{Pre}\left(\overset{B}{\neg}\, \psi\right)\right)$. The result follows from Lemma 5.1 and the fact that regular languages are closed under complementation. □

From Theorem 4.3 and Lemma 5.3 we get the following

Theorem 5.4 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 *effective* if for each configuration c we can compute the set $Post(c)$. Observe that this implies that the game is finitely branching.

Suppose that we want to check whether a configuration $c_{init} \in C_A$ is winning. 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 $c \in C_A$, $c \notin C_F$, and there is a predecessor of the 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 effective.

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. The algorithm answers “yes” if and only if the resulting tree evaluates positively.

Theorem 6.1 The safety problem is decidable for effective A-downward closed games.

A-LCS An *A-LCS* has the same syntax as a B-LCS. 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 effective. 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 give a characterization of the set of winning configurations as we did for B-LCS. By a similar reasoning to Lemma 3.1, the set $Pre(\overset{B}{\neg} E_i)$ is upward closed and therefore can be characterized by a finite set of upward closed constraints for each $i \geq 0$. In turn, the set $\bigcup_{i\geq 0} D_i$ can be characterized by a finite set of negation constraints. We show that we cannot compute a finite set of negation constraints ψ such that $[\![\psi]\!] = \bigcup_{i\geq 0} D_i$, as follows.

We reduce an uncomputability result reported in [BM99] for transition systems induced by lossy channel systems. The results in [BM99] imply that we cannot characterize the set of configurations c satisfying the property $c \models \exists_\infty \Box \neg S_F$, i.e., we cannot characterize 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 (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 in $\mathcal{L}'$ if and only if $c \models \exists_\infty \Box \neg S_F$.

From this, we get the following:

Theorem 6.3 We cannot compute a finite set of negation constraints characterizing the set of non-winning configurations in an A-LCS (although such a set always exists).

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*. 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.
- $C_A = \{(s, x, y) \mid s \in S_A\}$.
- $C_B = \{(s, x, y) \mid s \in S_B\}$.
- $(s_1, x_1, y_1) \longrightarrow (s_2, x_2, y_2)$ iff $(s_1, (a, b), s_2) \in T$, and $x_2 = x_1 + a$, 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.
- $C_F = \{(s, x, y) \mid s \in S_F\}$.
- $(s_1, x_1, y_1) \preceq (s_2, x_2, y_2)$ 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 $x_2 = x_1 + a$, and $y_2 = y_1 + b$; or
- $(s_1, x = 0?, s_2) \in T_M$ and $x_2 = x_1 = 0$, 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 $M = (S_M, T_M)$ and two states $s_{init}, s_f \in S_M$.

Question Is there a sequence

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

of transitions such that $s_0 = s_{init}$, $x_0 = 0$, $y_0 = 0$, 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 $M = (S_M, T_M)$ and two states $s_{init}, s_f \in S_M$, we construct a corresponding VASS game, such that the reachability problem has a positive answer if and only if the game problem has a negative answer. Intuitively, player B emulates the moves of the 2-counter machine, 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. 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. More formally, we define the VASS game $\mathcal{V} = (S, S_A, S_B, T, S_F)$ as follows:

- $S_A = \left\{ s_t^A \mid t \in T_M \right\} \cup \left\{ s_*^A, s_{reached}^A, s_{init}^A \right\}$. 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.
- $S_B = \left\{ s^B \mid s \in S_M \right\} \cup \left\{ s_*^B \right\}$. 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 $(s_1, (a, b), s_2) \in T_M$, there are two transitions in T, namely $\left(s_1^B, (a, b), s_t^A\right)$ and $\left(s_t^A, (0, 0), s_2^B\right)$. Player B chooses a move, and player A follows passively.
- For each transition t of the form $(s_1, x = 0?, s_2) \in T_M$, there are three transitions in T, namely $\left(s_1^B, (0, 0), s_t^A\right)$, $\left(s_t^A, (0, 0), s_2^B\right)$, and $\left(s_t^A, (-1, 0), s_*^B\right)$. 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 $(s_1, y = 0?, s_2) \in T_M$ are handled in a similar manner to the previous case.
- There are five additional transitions in T, namely an initializing transition $\left(s_{init}^A, (0, 0), s_{init}^B\right)$; an escape loop to detect that the final state has been

reached $\left(s_f^B, (0,0), s_{reached}^A\right)$ and $\left(s_{reached}^A, (0,0), s_f^B\right)$; a loop to detect illegal moves $\left(s_*^B, (0,0), s_*^A\right)$ and $\left(s_*^A, (0,0), s_*^B\right)$.

- $S_F = \left\{s_{reached}^A\right\}$.

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

$(s_0, x_0, y_0) \longrightarrow (s_1, x_1, y_1) \longrightarrow (s_2, x_2, y_2) \longrightarrow \cdots \longrightarrow (s_n, x_n, y_n)$ of transitions in M with $s_0 = s_{init}$, $x_0 = 0$, $y_0 = 0$, and $s_n = s_f$ iff the configuration $\left(s_{init}^A, 0, 0\right)$ is not winning in G.

(if) Suppose that $\left(s_{init}^A, 0, 0\right)$ is not winning in G. Then for any A-strategy, there exists a B-strategy such that $P\left(\left(s_{init}^B, 0, 0\right), \sigma_A, \sigma_B\right)$ is not winning. 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 $t = (s_1, (a,b), s_2)$ then $c = \left(s_2^B, x, y\right)$.
- if $t = (s_1, x = 0?, s_2)$ and $x > 0$ then $c = \left(s_*^B, x-1, y\right)$.
- if $t = (s_1, x = 0?, s_2)$ and $x = 0$ then $c = \left(s_2^B, x, y\right)$.
- the cases for $t = (s_1, y = 0?, s_2)$ are defined in a similar manner as the two previous cases.

Consider $P\left(\left(s_{init}^B, 0, 0\right), \sigma_A, \sigma_B\right) = c_0, c_1, c_2, \ldots$. Since $P\left(\left(s_{init}^B, 0, 0\right), \sigma_A, \sigma_B\right)$ is not winning for some B-strategy σ_B, there is a $k : 0 \leq k \leq \mathit{length}(P)$ such that c_k is of the form $\left(s_{reached}^A, x, y\right)$. 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 $\left(s_*^A, x, y\right)$ or $\left(s_*^B, x, y\right)$ (otherwise the definition of σ_A would imply that $P\left(\left(s_{init}^A, 0, 0\right), \sigma_A, \sigma_B\right)$ is winning). This means that $P\left(\left(s_{init}^A, 0, 0\right), \sigma_A, \sigma_B\right)$ is in fact of the form

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

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 $P(\left(s_{init}^B, 0, 0\right), \sigma_A, \sigma_B)$ is not winning for each A-strategy σ_A. We define σ_B to be any B-strategy such that

$\sigma_B\big((s_i^B, x_i, y_i)\big) = \big(s_{t_i}^A, x_i + a_i, y_i + b_i)\big) = \big(s_{t_i}^A, x_{i+1}, y_{i+1})\big)$, for each $i : 0 \leq i < n$. We observe that for any A-strategy we have $\sigma_A\big((s_{t_i}^A, x_{i+1}, y_{i+1})\big) = \big(s_{i+1}^B, x_{i+1}, y_{i+1}\big)$. This follows from the fact the transitions to s_*^B are never enabled during the above sequence.

8 Parity Games

A *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 *parity winning* if $r(P)$ is even. We say that c is *parity winning* 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 parity winning.

The parity problem

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

Question Is c (parity) winning?

We show below that the 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 parity problem is undecidable for A-LCS games.

In [AJ96a] we show undecidability of the *recurrent state problem*, for transition systems based on lossy channel systems.

The recurrent state 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 recurrent state problem for LCS to the 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 $\big(s_1^A, op, s_t^B\big)$ and $\big(s_t^B, nop, s_2^A\big)$ 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 $\left(s_3^*, nop, s_{init}^A\right)$. This is to get to the initial state of $\mathcal{L}$ when the channel content is ready.
- A transition $\left(s^A, nop, s_5^*\right)$ 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:

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

We show that (s_1^*, ϵ) is parity-winning 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

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

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:

- $\sigma_A\left(\left(s_1^*, w^i\right)\right) = \left(s_2^*, w^{i+1}\right)$, 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};
- $\sigma_A(\left(s_i^A, w_i\right)) = \left(s_{t_i}^B, w_{i+1}\right)$ 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 $P\left((s_1^*, \epsilon), \sigma_A, \sigma_B\right)$ stays indefinitely long in the rank 2 zone. Thus, the rank of the entire play is 2, and the initial configuration is parity-winning.

(only if) Let us assume that (s_1^*, ϵ) is parity-winning for the A-strategy σ_A. By construction of $\mathcal{L}'$, we know that the rank of a play $P\left((s_1^*, \epsilon), \sigma_A, \sigma_B\right)$, 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 $(s_3^*, w_0), \left(s_{init}^A, w_0\right), (s_1, w_1), \ldots$.

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

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

is a valid sequence of transitions of $\mathcal{L}$, and it is infinite.

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 [AČJYK00] 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 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 the simple class of parity conditions (implying undecidability also in the case of the standard definition). Using results in [May00], 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 4 (configurations have ranks 0, 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

[AČJYK00] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160:109–127, 2000.

[AD90] R. Alur and D. Dill. Automata for modelling real-time systems. In *Proc. ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335, 1990.

[AHK97] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc.* 38^{th} *Annual Symp. Foundations of Computer Science*, pages 100–109, 1997.

[AJ96a] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[AJ96b] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In *Proc. ICATPN'2001: 22nd Int. Conf. on application and theory of Petri nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 53 –70, 2001.

[BBK77] J. M. Barzdin, J. J. Bicevskis, and A. A. Kalninsh. Automatic construction of complete sample systems for program testing. In *IFIP Congress, 1977*, 1977.

[BM99] A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *Symp. on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 323–333, 1999.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In Abiteboul and Shamir, editors, *Proc. ICALP '94,* 21^{st} *International Colloquium on Automata, Languages, and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer Verlag, 1994.

[Del00] G. Delzanno. Automatic verification of cache coherence protocols. In Emerson and Sistla, editors, *Proc.* 12^{th} *Int. Conf. on Computer Aided*

Verification, volume 1855 of *Lecture Notes in Computer Science*, pages 53–68. Springer Verlag, 2000.

[DEP99] G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In *Proc. CSL'99*, 1999.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. LICS' 99* 14th *IEEE Int. Symp. on Logic in Computer Science*, 1999.

[Fin94] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.

[FS01] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

[LdAR01] T. Henzinger L. de Alfaro and R.Majumdar. Symbolic algorithms for infinite state games. In *Proc. CONCUR 2001,* 12th *Int. Conf. on Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, 2001.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*, number 1776 in Lecture Notes in Computer Science, 2000.

[RSB04] J.-F. Raskin, Mathias Samuelides, and Laurent Van Begin. Games for counting abstractions., 2004. To appear in the proceedings of AVOCS04, ENTCS.

[Tho02] W. Thomas. Infinite games and verification. In *Proc.* 14th *Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–64, 2002.