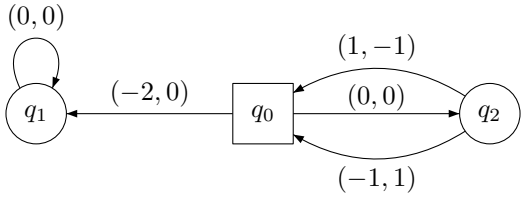
2.9 Verification of dynamic and parameterized systems 2011

Lecture 2: Monotonic Games Laurent Doyen

2.1. Monotonic Games. We drop the assumption that Q is a finite set[1]. Let $\preceq \subseteq (Q_1 \times Q_1) \cup (Q_2 \times Q_2)$ be a decidable well-quasi order. A game $\langle Q, E\rangle$ is *monotonic* with respect to $\preceq$ if for all $q_1, q_2, q_3 \in Q$, if $q_1 \preceq q_2$ and $(q_1, q_3) \in E$, then there exists $q_4 \in Q$ such that $q_3 \preceq q_4$ and $(q_2, q_4) \in E$.

Games on VASS (Vector-Addition System with States) induce monotonic games. A game on VASS consists of a game graph $\langle Q, E\rangle$ and a function $w : E \to \mathbb{Z}^d$ where d is the number of counters. It induces an infinite game graph $\langle Q \times \mathbb{N}^d, E'\rangle$ where $(q, v), (q', v') \in E'$ if $(q, q') \in E$ and $v' = v + w(q, q')$. Note the implicit requirement that $v' \geq 0$ (and $v \geq 0$).



VASS games can be used to simulate 2-counter Minsky machines 2CM. A 2-*counter machine* M consists of a finite set of control states Q, an initial state $q_I \in Q$, a final state $q_F \in Q$, a set C of counters ($|C| = 2$) and a finite set δ_M of instructions manipulating two integer-valued counters. Instructions are of the form

$q :\ c := c + 1$ **goto** q'

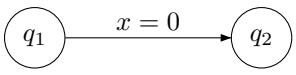
$q :$ **if** $c = 0$ **then goto** q' **else** $c := c - 1$ **goto** q''.

Formally, instructions are tuples (q, α, c, q') where $q, q' \in Q$ are source and target states respectively, the action $\alpha \in \{inc, dec, 0?\}$ applies to the counter $c \in C$. We assume that M is deterministic: for every state $q \in Q$, either there is exactly one instruction $(q, \alpha, \cdot, \cdot) \in \delta_M$ and $\alpha = inc$, or there are two instructions $(q, dec, c, \cdot), (q, 0?, c, \cdot) \in \delta_M$.

A *configuration* of M is a pair (q, v) where $q \in Q$ and $v : C \to \mathbb{N}$ is a valuation of the counters. An *accepting run* of M is a finite sequence $\pi = (q_0, v_0)\delta_0(q_1, v_1)\delta_1 \dots \delta_{n-1}(q_n, v_n)$ where $\delta_i = (q_i, \alpha_i, c_i, q_{i+1}) \in \delta_M$ are instructions and (q_i, v_i) are configurations of M such that $q_0 = q_I$, $v_0(c) = 0$ for all $c \in C$, $q_n = q_F$, and for all $0 \leq i < n$, we have $v_{i+1}(c) = v_i(c)$ for $c \neq c_i$, and (a) if $\alpha = inc$, then $v_{i+1}(c_i) = v_i(c_i) + 1$ (b) if $\alpha = dec$, then $v_i(c_i) \neq 0$ and $v_{i+1}(c_i) = v_i(c_i) - 1$, and (c) if $\alpha = 0?$, then $v_{i+1}(c_i) = v_i(c_i) = 0$. The corresponding *run trace* of π is the sequence of instructions $\bar{\pi} = \delta_0\delta_1 \dots \delta_{n-1}$.

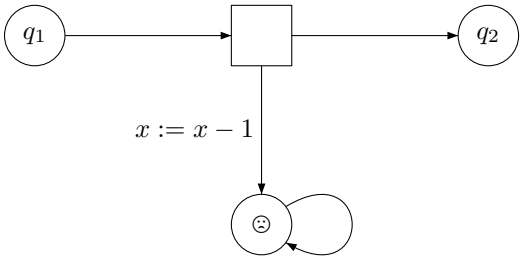
The *halting problem* is to decide, given a 2-counter machine M, whether M has an accepting run. This problem is undecidable.

In the simulation of 2CM by game on VASS, Player 1 simulates the execution of the 2CM, and player 2 checks the faithful simulation of zero tests. A transition with a zero-test on counter x

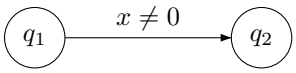


is replaced by the gadget:

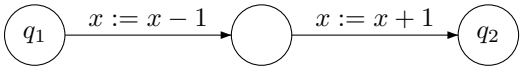
[1] However, we assume *finite branching* i.e., the set $E(q) = \{q' \mid (q, q') \in E\}$ is finite for all $q \in Q$.



And transitions with a non-zero test on counter x



are replaced by the gadget:



Edges to sink states are added to avoid deadlock states. Since the halting problem for 2CM is undecidable we have the following result.

Theorem 2A. Safety and reachability VASS games are undecidable.

2.2. Downward-closed Games. A game $\langle Q, E\rangle$ is $\preceq$-*downward-closed* if for all $q_1, q_2, q_3 \in Q$, if $q_1 \preceq q_2$ and $(q_1, q_3) \in E$, then $(q_2, q_3) \in E$. We assume that $E \subseteq (Q_1 \times Q_2) \cup (Q_2 \times Q_1)$, i.e. the players strictly alternate. Note that downward-closed games are monotonic.

Given a safety objective $\mathsf{Safe}(\mathcal{T})$, and an initial state q_0, consider the tree T with root labeled by q_0 constructed as follows. For each node $\mathtt{n}$ with label q in the tree:

> if $q \in Q \setminus \mathcal{T}$ then $\mathtt{n}$ is a leaf, declared *unsuccessful*.
>
> otherwise, if $q \in Q_1$ and an ancestor node of $\mathtt{n}$ has label $q' \preceq q$ then $\mathtt{n}$ is a leaf, declared *successful*
>
> otherwise, we expand the node $\mathtt{n}$, adding successors labeled by the set $\{q' \mid (q, q') \in E\}$ of successors of q.

Since $\preceq$ is a well-quasi order, this tree is finite by König's Lemma. The tree T defines a finite safety game where nodes labeled by states in Q_1 belong to player 1, and nodes labeled by states in Q_2 belong to player 2. The safety objective is to avoid unsuccessful leaves.

A winning strategy in a $\preceq$-downward-closed game can be mapped to a winning strategy in the tree T. And a winning strategy σ_{T} in the tree T can be mapped to a winning strategy σ in a $\preceq$-downward-closed game as follows.

By induction, we assume that the strategy σ is already defined for play prefixes of length $1, \ldots, k$ and we construct $\sigma(\rho)$ for play prefixes ρ of length $k+1$ as follows. We assume that ρ is compatible with the strategy σ. Consider a stack on which we push the states in ρ, and such that the content of the stack is always a path in T. Whenever we push state q corresponding to a leaf of T, there exists a state q' in the stack such that $q' \preceq q$. We remove all states above q' before pushing the next state. In this way, after processing ρ the stack contains a path in T, and we define $\sigma(\rho)$ as the state chosen by σ_{T} on this path. The strategy σ is well defined (why?) and since non-target states are avoided by σ_{T} in T, the strategy σ is winning for the safety objective $\mathsf{Safe}(\mathcal{T})$ from q_0.

Theorem 2B. Safety downward-closed games are decidable.