On the (high) undecidability of distributed program synthesis

David Janin

LaBRI, Université de Bordeaux I

January 21, 2007

Outlines

- Games and program synthesis
  - Program synthesis through games
  - The centralized programming case
  - The distributed programming case
- Expressiveness of distributed games
  - A model of computation : domino games
  - Dominos and distributed games
  - Within the arithmetical hierarchy
  - Above the arithmetical hierarchy
- Conclusion

Outlines

Games and program synthesis

- Program synthesis through games
- The centralized programming case
- The distributed programming case

Expressiveness of distributed games

- A model of computation : domino games
- Dominos and distributed games
- Within the arithmetical hierarchy
- Above the arithmetical hierarchy

Conclusion

Outlines

Designing (correct) programs with games

Designing correct programs

Goal: Given a program spec. S, design a program P s.t. $P \models S$.

The game reduction

Compute a game $\mathcal{G}_S$ and a mapping $P \mapsto \sigma_P$ that maps (correct) programs P onto (winning) strategies σ_P. Then, designing a (correct) program amounts to finding a (winning) strategy.

A fundationnal approach ?

Up to (arbitrary !) game reductions programs are (winning) strategies. The game approach is thus fairly universal.

Designing (correct) programs with games

Designing correct programs

Goal: Given a program spec. S, design a program P s.t. $P \models S$.

The game reduction

Compute a game $\mathcal{G}_S$ and a mapping $P \mapsto \sigma_P$ that maps (correct) programs P onto (winning) strategies σ_P. Then, designing a (correct) program amounts to finding a (winning) strategy.

A fundationnal approach ?

Up to (arbitrary !) game reductions programs are (winning) strategies. The game approach is thus fairly universal.

Designing (correct) programs with games

Designing correct programs

Goal: Given a program spec. S, design a program P s.t. $P \models S$.

The game reduction

Compute a game $\mathcal{G}_S$ and a mapping $P \mapsto \sigma_P$ that maps (correct) programs P onto (winning) strategies σ_P. Then, designing a (correct) program amounts to finding a (winning) strategy.

A fundationnal approach ?

Up to (arbitrary !) game reductions programs are (winning) strategies. The game approach is thus fairly universal.

Designing (correct) programs with games

Designing correct programs

Goal: Given a program spec. S, design a program P s.t. $P \models S$.

The game reduction

Compute a game $\mathcal{G}_S$ and a mapping $P \mapsto \sigma_P$ that maps (correct) programs P onto (winning) strategies σ_P. Then, designing a (correct) program amounts to finding a (winning) strategy.

A fundationnal approach ?

Up to (arbitrary !) game reductions programs are (winning) strategies. The game approach is thus fairly universal.

Designing (correct) programs with games

Designing correct programs

Goal: Given a program spec. S, design a program P s.t. $P \models S$.

The game reduction

Compute a game $\mathcal{G}_S$ and a mapping $P \mapsto \sigma_P$ that maps (correct) programs P onto (winning) strategies σ_P. Then, designing a (correct) program amounts to finding a (winning) strategy.

A fundationnal approach ?

Up to (arbitrary !) game reductions programs are (winning) strategies. The game approach is thus fairly universal.

The Two Player Game Setting

A one against one game

- the Process player (Smiley),
- the Environment player (Fred).

The Two Player Game Setting

A one against one game

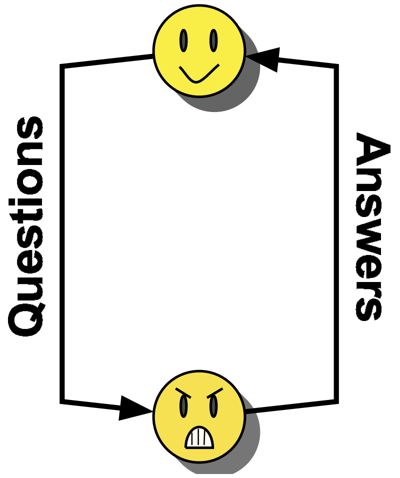
- the Process player (Smiley),
- the Environment player (Fred).

The Two Player Game Setting

A one against one game

- the Process player (Smiley),
- the Environment player (Fred).

Two players games definition

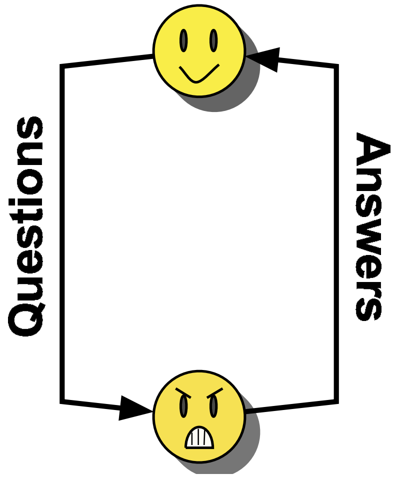


A game $\mathcal{G} = \langle\rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when:
  either $\sigma * \tau \in (A.Q)^+$ (finite case)
  or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case)
  W is the infinitary winning condition.

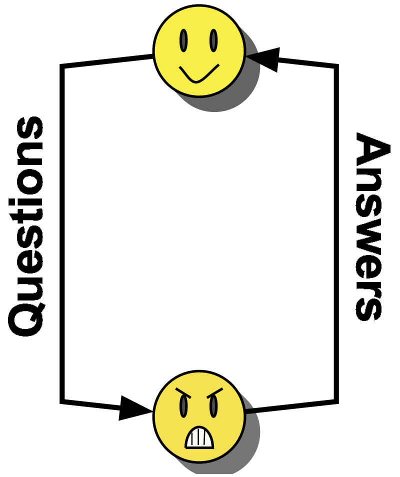
Two players games definition

A game $\mathcal{G} = \langle Q, A \rangle$



- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when: either $\sigma * \tau \in (A.Q)^+$ (finite case) or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case) W is the infinitary winning condition.

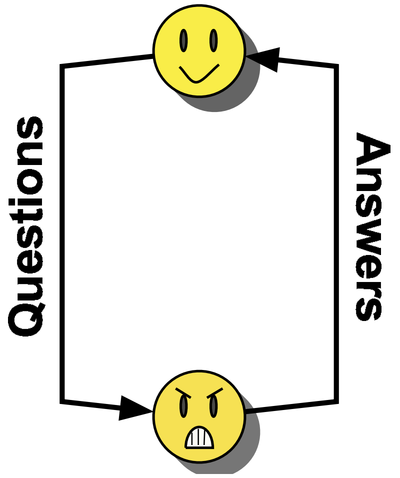
Two players games definition



A game $\mathcal{G} = \langle Q, A, a_0 \rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when: either $\sigma * \tau \in (A.Q)^+$ (finite case) or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case) W is the infinitary winning condition.

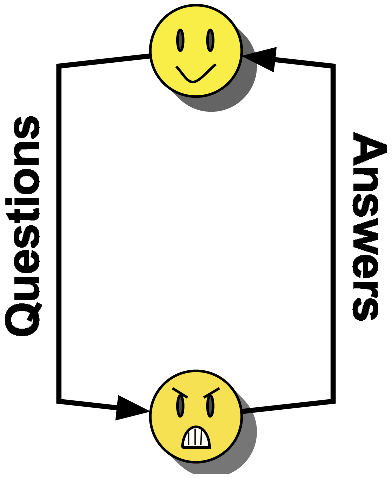
Two players games definition



A game $\mathcal{G} = \langle Q, A, a_0, R_P, R_E \rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when: either $\sigma * \tau \in (A.Q)^+$ (finite case) or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case) W is the infinitary winning condition.

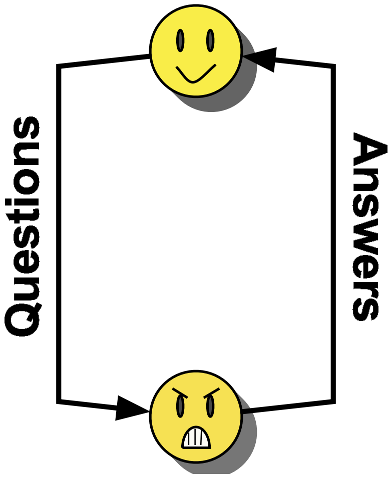
Two players games definition



A game $\mathcal{G} = \langle Q, A, a_0, R_P, R_E \rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when: either $\sigma * \tau \in (A.Q)^+$ (finite case) or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case) W is the infinitary winning condition.

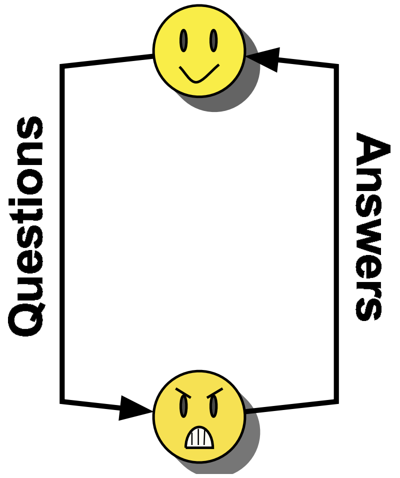
Two players games definition



A game $\mathcal{G} = \langle Q, A, a_0, R_P, R_E \rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when: either $\sigma * \tau \in (A.Q)^+$ (finite case) or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case) W is the infinitary winning condition.

Two players games definition



A game $\mathcal{G} = \langle Q, A, a_0, R_P, R_E, W \rangle$

- Questions Q (Env. pos.) and Answers A (Proc. pos.) with an initial fact $a_0 \in A$,
- Game Rules : $R_P \subseteq A \times Q$ and $R_E \subseteq Q \times A$,
- Env. Strategy $\tau : Q^* \to A$ (with $\tau_E(\epsilon) = a_0$) and Process's strategy $\sigma : A^* \to Q$
- Induced (maximal) play : $\sigma * \tau \in (A.Q)^*.A + (A.Q)^+ + (A.Q)^\omega$
- Process wins when:
  either $\sigma * \tau \in (A.Q)^+$ (finite case)
  or $\sigma * \tau \in W \subseteq (A.Q)^\omega$ (infinite case)
  W is the infinitary winning condition.

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$,
- Safety game when $W = A^\omega$,
- Parity game [McN,Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ such that
$$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ as above such that
$$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\}$$

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$,
- Safety game when $W = A^\omega$,
- Parity game [McN,Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ such that
$$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ as above such that
$$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\}$$
.

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$,
- Safety game when $W = A^\omega$,
- Parity game [McN,Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ such that
$$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ as above such that
$$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\} .$$

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$,
- Safety game when $W = A^\omega$,
- Parity game [McN,Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ such that
$$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ as above such that
$$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\}$$
.

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$, i.e. parity with range [1]
- Safety game when $W = A^\omega$, i.e. parity with range [0]
- Parity game [McN,Mos] with priority range $[m, n]$
  when there is $c : A \longrightarrow [m, n]$ such that
  $$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$
  when there is $c : A \longrightarrow [m, n]$ as above such that
  $$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\}$$.

Some classical infinitary conditions

Definition

Games $\mathcal{G}$ is a :

- Reachability game when $W = \emptyset$, i.e. parity with range [1]
- Safety game when $W = A^\omega$, i.e. parity with range [0]
- Parity game [McN,Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ such that
$$W = \{w \in (A.Q)^\omega : \liminf c \circ \pi_A(w) \equiv 0(2)\}$$
- Weak parity game [Mos] with priority range $[m, n]$ when there is $c : A \to [m, n]$ as above such that
$$W = \{w \in (A.Q)^\omega : c(w) \nearrow \wedge \lim c \circ \pi_A(w) \equiv 0(2)\}$$
.

Some facts

Fact

In a finite game $\mathcal{G} = \langle Q, A, a_0, R_S, R_F, W \rangle$ with ω-regular W:

- *Game determinacy: either Process or Environment player has a winnings strategy [Martin,EmeJut]*
- *Computability : winning strategies are computable,[BücLand]*
- *Complexity: reachability or safety games can be solve in linear time (and P-complete), weak parity games can be solved in polynomial time, solving parity game can be solve in exp. time (though in NP∩ co-NP) [EmeJut,Jur]*

Some facts

Fact

In a finite game $\mathcal{G} = \langle Q, A, a_0, R_S, R_F, W\rangle$ with ω-regular W:

- *Game determinacy: either Process or Environment player has a winnings strategy [Martin,EmeJut]*
- *Computability : winning strategies are computable,[BücLand]*
- *Complexity: reachability or safety games can be solve in linear time (and P-complete), weak parity games can be solved in polynomial time, solving parity game can be solve in exp. time (though in NP$\cap$ co-NP) [EmeJut,Jur]*

Some facts

Fact

In a finite game $\mathcal{G} = \langle Q, A, a_0, R_S, R_F, W \rangle$ with ω-regular W:

- *Game determinacy: either Process or Environment player has a winnings strategy [Martin,EmeJut]*
- *Computability : winning strategies are computable,[BücLand]*
- *Complexity: reachability or safety games can be solve in linear time (and P-complete), weak parity games can be solved in polynomial time, solving parity game can be solve in exp. time (though in NP∩ co-NP) [EmeJut,Jur]*

Some facts

Fact

In a finite game $\mathcal{G} = \langle Q, A, a_0, R_S, R_F, W\rangle$ with ω-regular W:

- *Game determinacy: either Process or Environment player has a winnings strategy [Martin,EmeJut]*
- *Computability : winning strategies are computable,[BücLand]*
- *Complexity: reachability or safety games can be solve in linear time (and P-complete), weak parity games can be solved in polynomial time, solving parity game can be solve in exp. time (though in NP$\cap$ co-NP) [EmeJut,Jur]*

Some facts

Fact

In a finite game $\mathcal{G} = \langle Q, A, a_0, R_S, R_F, W \rangle$ with ω-regular W:

- *Game determinacy: either Process or Environment player has a winnings strategy [Martin,EmeJut]*
- *Computability : winning strategies are computable,[BücLand]*
- *Complexity: reachability or safety games can be solve in linear time (and P-complete), weak parity games can be solved in polynomial time, solving parity game can be solve in exp. time (though in NP$\cap$ co-NP) [EmeJut,Jur]*

The Distributed Game Setting

An *n* against one player game

Many players called Processes against another player called Environment !

The Distributed Game Setting

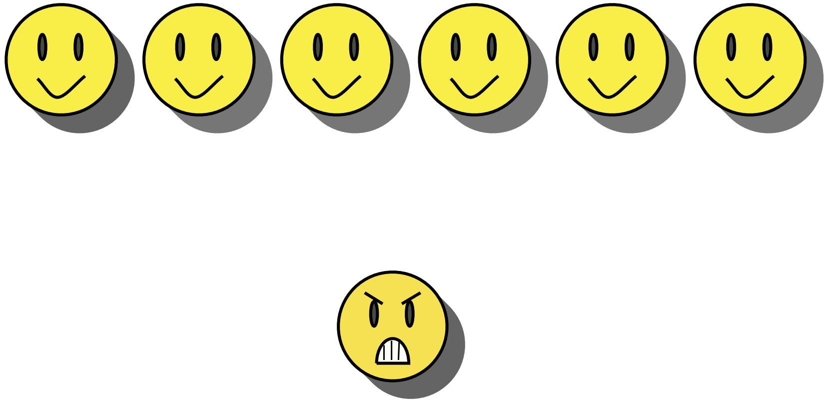
An *n* against one player game

Many players called Processes against another player called Environment !

The Distributed Game Setting

An *n* against one player game

Many players called Processes against another player called Environment !



Distributed games definition

Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes: $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n.$

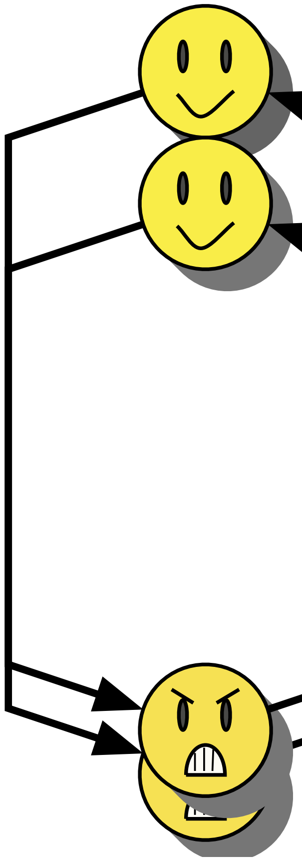
Distributed games definition

Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes: $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$.

Distributed games definition

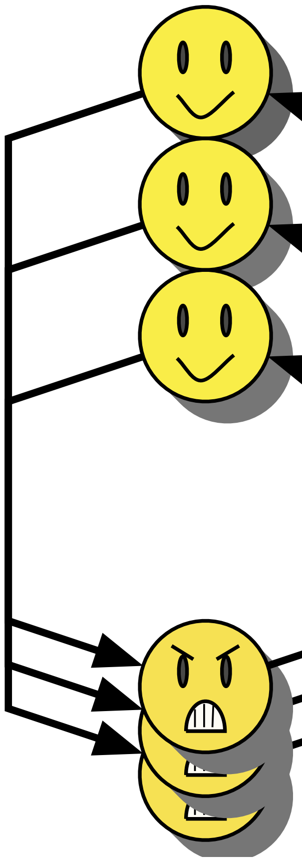


Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes:
  $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$.

Distributed games definition

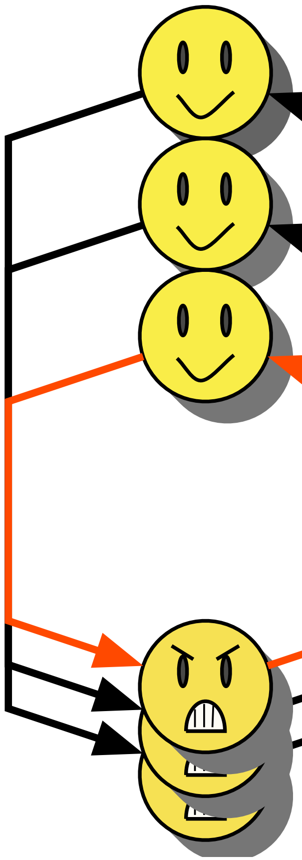


Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes: $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$.

Distributed games definition

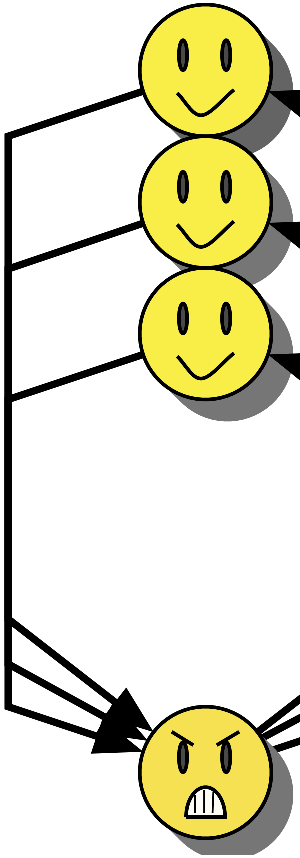


Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes: $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$.

Distributed games definition

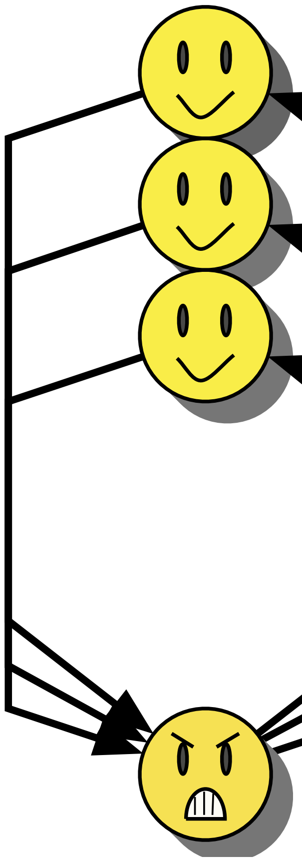


Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes:
  $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n.$

Distributed games definition

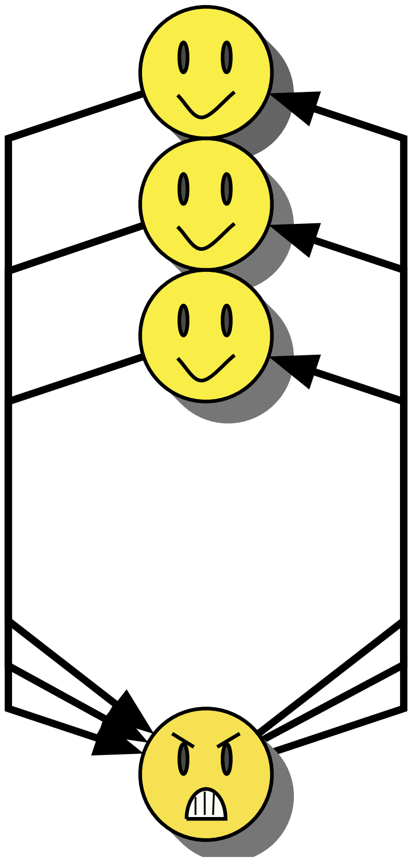


Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes:
  $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n.$

Distributed games definition



Features

- Many local games $\mathcal{G}_i = \langle Q_i, A_i, a_i, R_{i,S}, R_{i,F} \rangle$
- with partial informations for the many Processes:
  $R_{i,S} \subseteq A_i \times Q_i$
- and total information for the unique Environment
- but possibly restricted moves for Environment $R_F \subseteq R_{1,F} \otimes \cdots \otimes R_{n,F}$

$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$.

Distributed plays and strategies

Definition

Given a distributed game

$$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$$

with $Q = \prod_{i \in [1,n]} Q_i$ and $A = \prod_{i \in [1,n]} A_i$, solving distributed game $\mathcal{G}$ amounts to finding local strategies $\{\sigma_i : A_i^* \to Q_i\}_{i \in [1,n]}$ such that, the induced distributed global strategy $\sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ defined, for every $w \in A^*$ by

$$(\sigma_1 \otimes \cdots \otimes \sigma_n)(w) = (\sigma_i \circ \pi_{A_i}(w))_{i \in [1,n]}$$

is winning in $\mathcal{G}$ w.r.t. a global winning condition W.

Distributed plays and strategies

Definition

Given a distributed game

$$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$$

with $Q = \prod_{i\in[1,n]} Q_i$ and $A = \prod_{i\in[1,n]} A_i$, solving distributed game $\mathcal{G}$ amounts to finding local strategies $\{\sigma_i : A_i^* \to Q_i\}_{i\in[1,n]}$ such that, the induced distributed global strategy $\sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ defined, for every $w \in A^*$ by

$$(\sigma_1 \otimes \cdots \otimes \sigma_n)(w) = (\sigma_i \circ \pi_{A_i}(w))_{i\in[1,n]}$$

is winning in $\mathcal{G}$ w.r.t. a global winning condition W.

Distributed plays and strategies

Definition

Given a distributed game

$$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$$

with $Q = \prod_{i \in [1,n]} Q_i$ and $A = \prod_{i \in [1,n]} A_i$, solving distributed game $\mathcal{G}$ amounts to finding local strategies $\{\sigma_i : A_i^* \to Q_i\}_{i \in [1,n]}$ such that, the induced distributed global strategy $\sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ defined, for every $w \in A^*$ by

$$(\sigma_1 \otimes \cdots \otimes \sigma_n)(w) = (\sigma_i \circ \pi_{A_i}(w))_{i \in [1,n]}$$

is winning in $\mathcal{G}$ w.r.t. a global winning condition W.

Distributed plays and strategies

Definition

Given a distributed game

$$\mathcal{G} \subseteq \mathcal{G}_1 \otimes \mathcal{G}_2 \otimes \cdots \otimes \mathcal{G}_n$$

with $Q = \prod_{i \in [1,n]} Q_i$ and $A = \prod_{i \in [1,n]} A_i$, solving distributed game $\mathcal{G}$ amounts to finding local strategies $\{\sigma_i : A_i^* \to Q_i\}_{i \in [1,n]}$ such that, the induced distributed global strategy $\sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ defined, for every $w \in A^*$ by

$$(\sigma_1 \otimes \cdots \otimes \sigma_n)(w) = (\sigma_i \circ \pi_{A_i}(w))_{i \in [1,n]}$$

is winning in $\mathcal{G}$ w.r.t. a global winning condition W.

Some facts

Fact

- *Game determininacy: Distributed games are not determined,*
- *Computability: Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*

Some facts

Fact

- *Game determininacy: Distributed games are not determined,*
- *Computability: Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*

Some facts

Fact

- *Game determininacy: Distributed games are not determined, hint : find a game winning for Processes but with no distributed winning strategy,*
- *Computability: Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*

Some facts

Fact

- *Game determininacy: Distributed games are not determined,*
- *Computability: Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*

Some facts

Fact

- *Game determininacy: Distributed games are not determined,*
- *Computability: Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*

Some facts

Fact

- *Game determininacy:* *Distributed games are not determined,*
- *Computability:* *Distributed games are partial information games and, as such are, in general, undecidable [PetRei80],*
- *Decidable sub-cases : E-deterministic or hierarchical distributed games game are decidable [PetRei80, PnuRos90, MohWal01]*
  hint : when one players "knows" other players positions the game simplifies

How distributed game are undecidable ?

For what “minimal” features ?

Open question till today : are finite two Processes distributed games decidable ?

How distributed game are undecidable ?
For what "minimal" features ?
Open question till today : are finite two Processes distributed games decidable ?

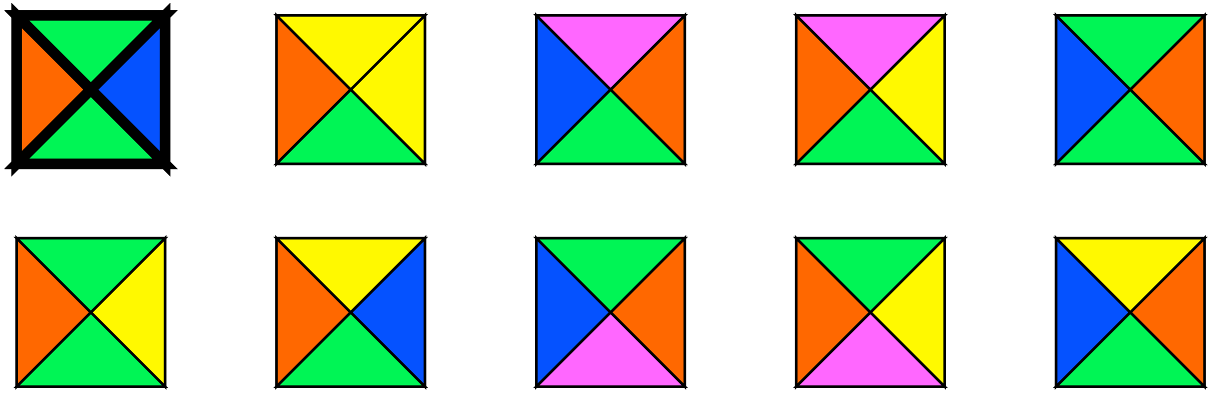
How distributed game are undecidable ?
For what “minimal” features ?
Open question till today : are finite two Processes distributed games decidable ?

Solitaire domino games (tiling systems)

Solitaire domino games (tiling systems)

Dominos

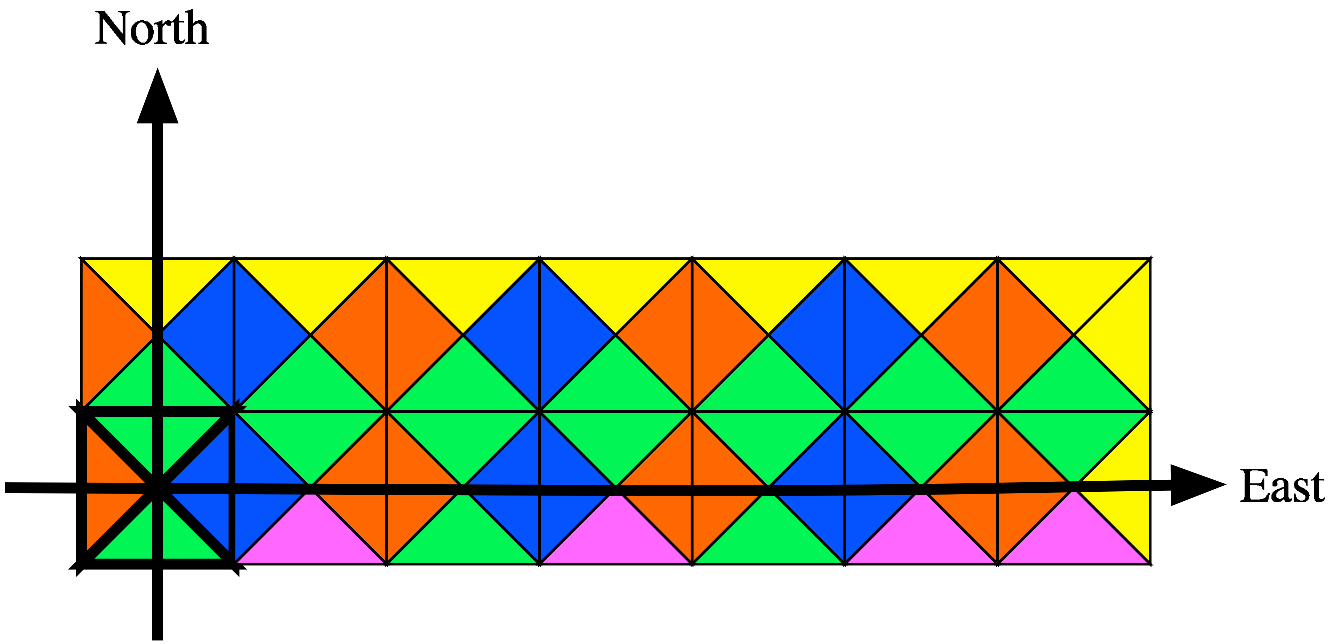
A set of dominos (or tiles) with a distinguished initial tile:



Solitaire domino games (tiling systems)

A finite tiling

(with yellow border color):



Definition

Given $D = \{n, s, w, e\}$, given a set of color C, given a set $T \subseteq (D \to Q \cup \{\#\})$ of tiles, with distinguished initial tile t_0, a T-tilling is a mapping $t : IN \times IN \to$ such that for every $(i, j) \in IN \times IN$

- **Initial condition**: $t(0,0) = t_0$,
- **E/W-compatibility**: $t(i,j)(e) = \#$ or $t(i,j)(e) = t(i,j+1)(w)$,
- **N/S-compatibility**: $t(i,j)(n) = \#$ or $t(i,j)(n) = t(i+1)(s)$,

Theorem (Harel et al.)

The set of finite (resp. infinite) domino tiling is Σ^0_1-complete (resp. Π^0_1-complete).

Hint

There is a correspondence between (finite domain) dominos tiling and TM accepting runs on the empty strings.

Definition

Given $D = \{n, s, w, e\}$, given a set of color C, given a set $T \subseteq (D \rightarrow Q \cup \{\#\})$ of tiles, with distinguished initial tile t_0, a T-tilling is a mapping $t : I\!N \times I\!N \rightarrow$ such that for every $(i, j) \in I\!N \times I\!N$

- **Initial condition**: $t(0, 0) = t_0$,
- **E/W-compatibility**: $t(i, j)(e) = \#$ or $t(i, j)(e) = t(i, j + 1)(w)$,
- **N/S-compatibility**: $t(i, j)(n) = \#$ or $t(i, j)(n) = t(i + 1)(s)$,

Theorem (Harel et al.)

The set of finite (resp. infinite) domino tiling is Σ^0_1-complete (resp. Π^0_1-complete).

Hint

There is a correspondence between (finite domain) dominos tiling and TM accepting runs on the empty strings.

Definition

Given $D = \{n, s, w, e\}$, given a set of color C, given a set $T \subseteq (D \to Q \cup \{\#\})$ of tiles, with distinguished initial tile t_0, a T-tilling is a mapping $t : IN \times IN \to$ such that for every $(i, j) \in IN \times IN$

- **Initial condition**: $t(0, 0) = t_0$,
- **E/W-compatibility**: $t(i, j)(e) = \#$ or $t(i, j)(e) = t(i, j + 1)(w)$,
- **N/S-compatibility**: $t(i, j)(n) = \#$ or $t(i, j)(n) = t(i + 1)(s)$,

Theorem (Harel et al.)

The set of finite (resp. infinite) domino tiling is Σ^0_1-complete (resp. Π^0_1-complete).

Hint

There is a correspondence between (finite domain) dominos tiling and TM accepting runs on the empty strings.

Definition

Given $D = \{n, s, w, e\}$, given a set of color C, given a set $T \subseteq (D \rightarrow Q \cup \{\#\})$ of tiles, with distinguished initial tile t_0, a T-tilling is a mapping $t : IN \times IN \rightarrow$ such that for every $(i, j) \in IN \times IN$

- **Initial condition**: $t(0, 0) = t_0$,
- **E/W-compatibility**: $t(i, j)(e) = \#$ or $t(i, j)(e) = t(i, j + 1)(w)$,
- **N/S-compatibility**: $t(i, j)(n) = \#$ or $t(i, j)(n) = t(i + 1)(s)$,

Theorem (Harel et al.)

The set of finite (resp. infinite) domino tiling is Σ^0_1-complete (resp. Π^0_1-complete).

Hint

There is a correspondence between (finite domain) dominos tiling and TM accepting runs on the empty strings.

Definition

Given $D = \{n, s, w, e\}$, given a set of color C, given a set $T \subseteq (D \rightarrow Q \cup \{\#\})$ of tiles, with distinguished initial tile t_0, a T-tilling is a mapping $t : IN \times IN \rightarrow$ such that for every $(i, j) \in IN \times IN$

- **Initial condition**: $t(0, 0) = t_0$,
- **E/W-compatibility**: $t(i, j)(e) = \#$ or $t(i, j)(e) = t(i, j + 1)(w)$,
- **N/S-compatibility**: $t(i, j)(n) = \#$ or $t(i, j)(n) = t(i + 1)(s)$,

Theorem (Harel et al.)

The set of finite (resp. infinite) domino tiling is Σ^0_1-complete (resp. Π^0_1-complete).

Hint

There is a correspondence between (finite domain) dominos tiling and TM accepting runs on the empty strings.

Encoding dominos games into distributed games ?

Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

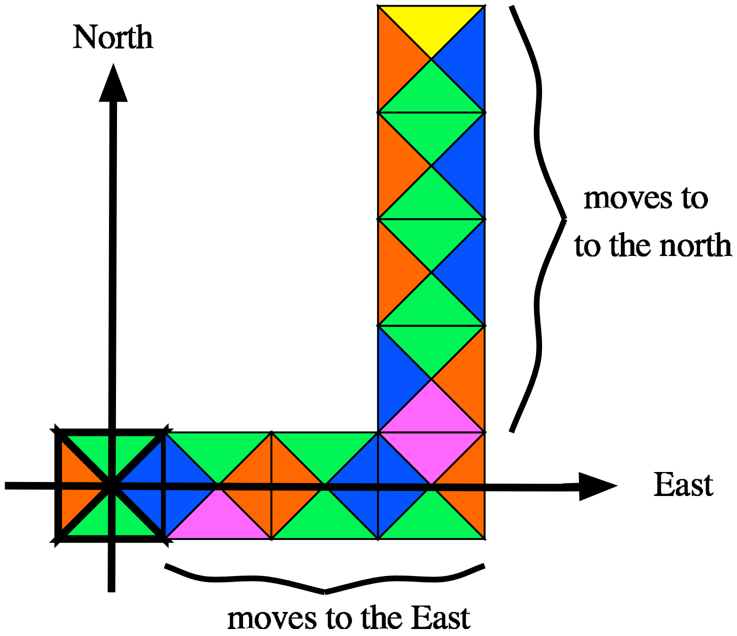
Pre-domino game

Pre-domino game

Define the two player game $\mathcal{G}_{T,t_0}$ where:

- Environment plays along $e^*.n^\omega$,
- Process answers by choosing tiles in T,
- Game rules guarantee E/W-comp. on first line,and N/S-comp. on columns.

A play in a pre-domino game



Quasi-tilings

Lemma

In pre-domino games, Process strategies are quasi-tilings.

Quasi-tilings

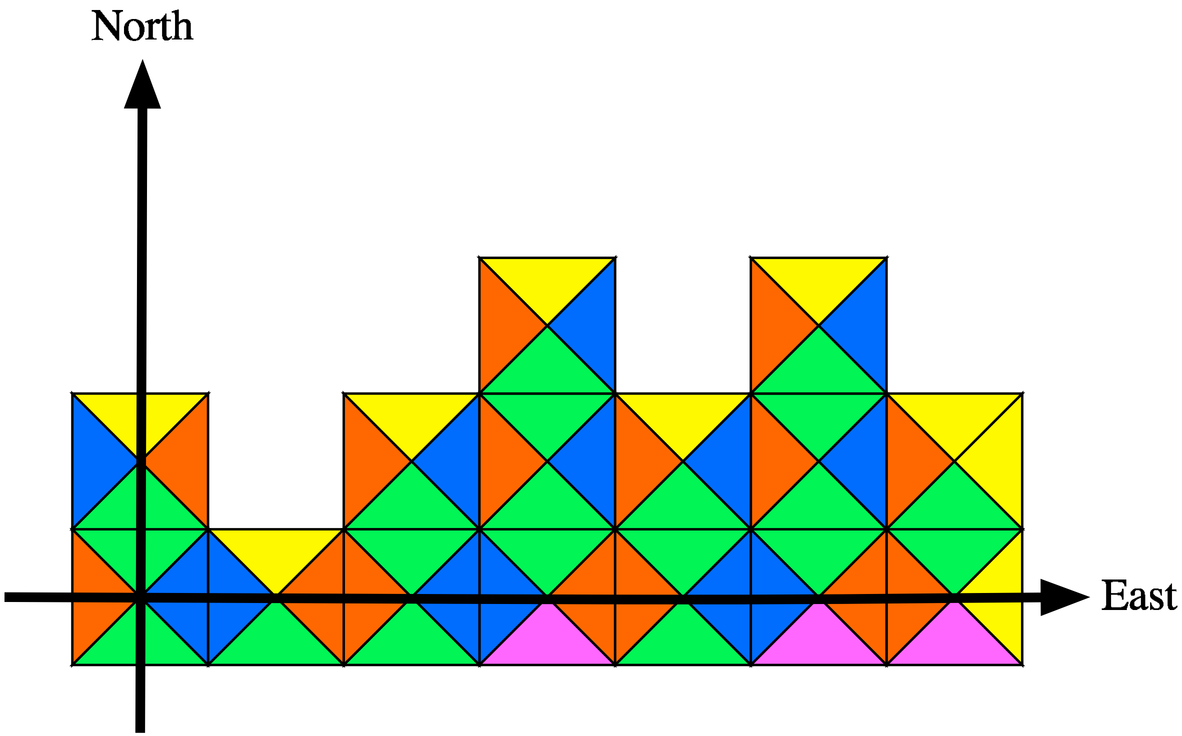
Lemma

In pre-domino games, Process strategies are quasi-tilings.

Quasi-tilings

Lemma

In pre-domino games, Process strategies are quasi-tilings.



Synchronizing two pre-domino games

A two-process distributed game

From two copies of the pre-domino game $\mathcal{G}_T$,

- Environment first chooses one extra bit (one per copy),
- and, accordingly:
  - checks local plays equality when bits are $(0, 0)$, $(0, 1)$ or $(1, 1)$,
  - checks local strategies E/W-compatibility when bits are $(1, 0)$,

Synchronizing two pre-domino games

A two-process distributed game

From two copies of the pre-domino game $\mathcal{G}_T$,

- Environment first chooses one extra bit (one per copy),
- and, accordingly:
  - checks local plays equality when bits are $(0,0)$, $(0,1)$ or $(1,1)$,
  - checks local strategies E/W-compatibility when bits are $(1,0)$,

Synchronizing two pre-domino games

A two-process distributed game

From two copies of the pre-domino game $\mathcal{G}_T$,

- Environment first chooses one extra bit (one per copy),
- and, accordingly:
  - checks local plays equality when bits are $(0,0)$, $(0,1)$ or $(1,1)$,
  - checks local strategies E/W-compatibility when bits are $(1,0)$,

Synchronizing two pre-domino games

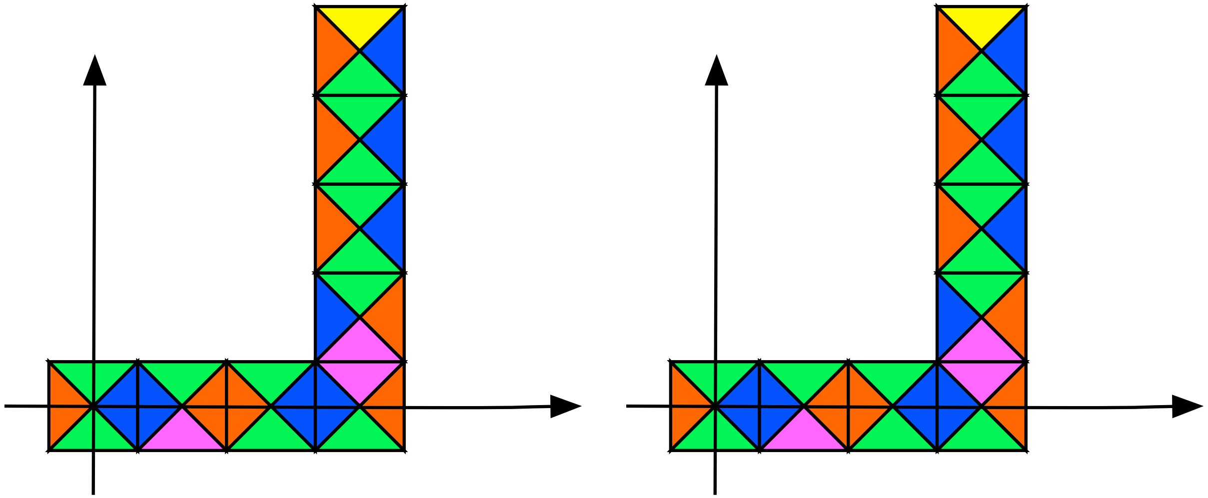
A two-process distributed game

From two copies of the pre-domino game $\mathcal{G}_T$,

- Environment first chooses one extra bit (one per copy),
- and, accordingly:
  - checks local plays equality when bits are $(0,0)$, $(0,1)$ or $(1,1)$,
  - checks local strategies E/W-compatibility when bits are $(1,0)$,

Forcing bits independence

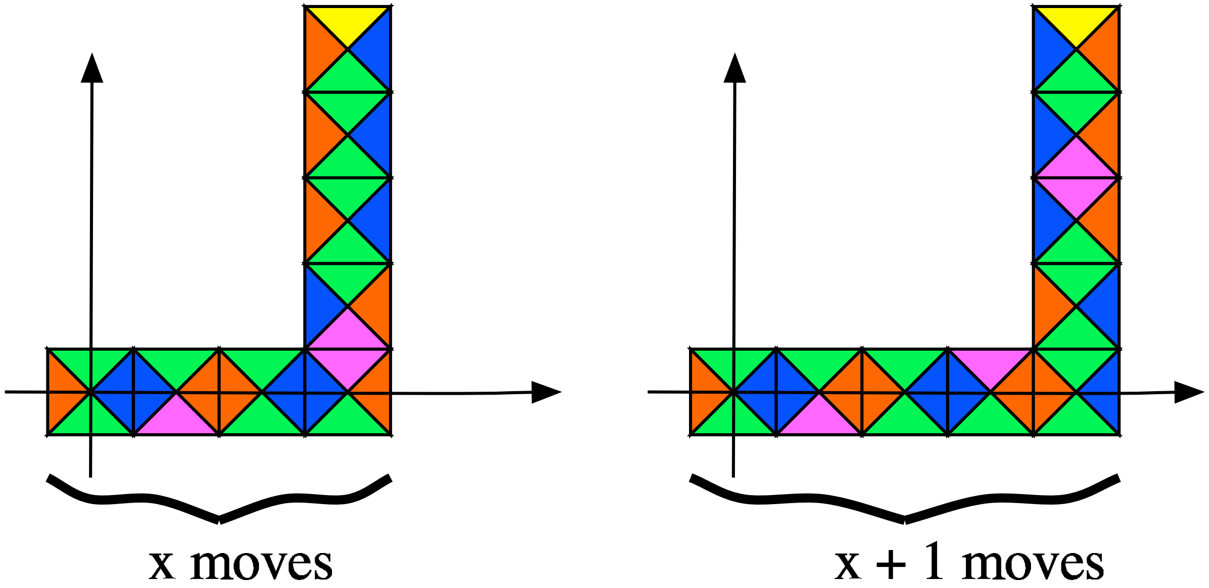
Local Process answers with bits $(0,0)$, $(0,1)$ or $(1,1)$ must be



equal.

Forcing E/W-compatibility

Local Process answers with bits $(1,0)$ must be E/W-compatible.



Lemma

In game $\mathcal{G} \subseteq \mathcal{G}_T \otimes \mathcal{G}_T$, both Process local (winning) strategies must be:

- *equals and extra bit independent since equals with bit values $(0,0)$, $(0,1)$, and $(1,1)$,*
- *E/W-compatible with bits value $(1,0)$,*

hence Process local (winning) strategies define tilings.

Lemma

In game $\mathcal{G} \subseteq \mathcal{G}_T \otimes \mathcal{G}_T$, both Process local (winning) strategies must be:

- *equals and extra bit independent*
  since equals with bit values $(0,0)$, $(0,1)$, and $(1,1)$,
- *E/W-compatible with bits value $(1,0)$,*

hence Process local (winning) strategies define tilings.

Lemma

In game $\mathcal{G} \subseteq \mathcal{G}_T \otimes \mathcal{G}_T$, both Process local (winning) strategies must be:

- *equals and extra bit independent*
  since equals with bit values $(0,0)$, $(0,1)$, *and* $(1,1)$,
- E/W*-compatible with bits value* $(1,0)$,

hence Process local (winning) strategies define tilings.

Corollary: reachability and safety cases

Theorem

Computing winning strategies in two processes distributed game with reachability (resp. safety) condition is Σ_1^0-complete (resp. Π_1^0-complete)

Proof.

(lower bound) Applies Harel results with tiling encodings. □

remark

This solve the open problem given in [MohWal01] refining [PetRei80] and [PnuRos90].

Corollary: reachability and safety cases

Theorem

Computing winning strategies in two processes distributed game with reachability (resp. safety) condition is Σ_1^0-complete (resp. Π_1^0-complete)

Proof.

(lower bound) Applies Harel results with tiling encodings. □

remark

This solve the open problem given in [MohWal01] refining [PetRei80] and [PnuRos90].

Corollary: reachability and safety cases

Theorem

Computing winning strategies in two processes distributed game with reachability (resp. safety) condition is Σ_1^0*-complete (resp.* Π_1^0*-complete)*

Proof.

(lower bound) Applies Harel results with tiling encodings. □

remark

This solve the open problem given in [MohWal01] refining [PetRei80] and [PnuRos90].

The number of Process' players does not change these results !

An the infinitary condition ?

The number of Process' players does not change these results !
An the infinitary condition ?

Definition (The arithmetical hierarchy)

A (finite word) predicate is Σ_0^0 or Π_0^0 when it is recursive.
A predicate is Σ_{n+1}^0 (resp. Π_{n+1}^0) when is of the form $\exists \vec{x} \varphi$ with $\varphi \in \Pi_n^0$ (resp. $\forall \vec{x} \varphi$ with $\varphi \in \Sigma_n^0$).

Theorem (Post)

The arithmetical hierarchy is strict. A predicates is Σ_{n+1}^0 if and only if it is definable by TM with Π_n^0 oracles.

Definition (The arithmetical hierarchy)

A (finite word) predicate is Σ_0^0 or Π_0^0 when it is recursive.
A predicate is Σ_{n+1}^0 (resp. Π_{n+1}^0) when is of the form $\exists \vec{x} \varphi$ with $\varphi \in \Pi_n^0$ (resp. $\forall \vec{x} \varphi$ with $\varphi \in \Sigma_n^0$).

Theorem (Post)

The arithmetical hierarchy is strict. A predicates is Σ_{n+1}^0 if and only if it is definable by TM with Π_n^0 oracles.

Definition (The arithmetical hierarchy)

A (finite word) predicate is Σ_0^0 or Π_0^0 when it is recursive.
A predicate is Σ_{n+1}^0 (resp. Π_{n+1}^0) when is of the form $\exists \vec{x} \varphi$ with $\varphi \in \Pi_n^0$ (resp. $\forall \vec{x} \varphi$ with $\varphi \in \Sigma_n^0$).

Theorem (Post)

The arithmetical hierarchy is strict. A predicates is Σ_{n+1}^0 if and only if it is definable by TM with Π_n^0 oracles.

Definition (The arithmetical hierarchy)

A (finite word) predicate is Σ_0^0 or Π_0^0 when it is recursive.
A predicate is Σ_{n+1}^0 (resp. Π_{n+1}^0) when is of the form $\exists \vec{x} \varphi$ with $\varphi \in \Pi_n^0$ (resp. $\forall \vec{x} \varphi$ with $\varphi \in \Sigma_n^0$).

Theorem (Post)

The arithmetical hierarchy is strict. A predicates is Σ_{n+1}^0 if and only if it is definable by TM with Π_n^0 oracles.

Definition (ω-ATM)

An ω-Alternating Turing Machine is an Alternating Turing Machine (ATM) possibly with infinite runs that are said accepting or not depending on a language of (accepting) infinite words W of control states. Parity and weak parity ω-ATM are defined accordingly.

Lemma

A languages $L \subseteq \Sigma^*$ is Σ^0_{n+1} (resp. Π^0_{n+1}) if and only if it is definable by means of an ω-ATM with weak parity condition of range $[1, n+1]$ (resp. $[0, n]$).

Definition (ω-ATM)

An ω-Alternating Turing Machine is an Alternating Turing Machine (ATM) possibly with infinite runs that are said accepting or not depending on a language of (accepting) infinite words W of control states. Parity and weak parity ω-ATM are defined accordingly.

Lemma

A languages $L \subseteq \Sigma^*$ is Σ^0_{n+1} (resp. Π^0_{n+1}) if and only if it is definable by means of an ω-ATM with weak parity condition of range $[1, n+1]$ (resp. $[0, n]$).

Definition (ω-ATM)

An ω-Alternating Turing Machine is an Alternating Turing Machine (ATM) possibly with infinite runs that are said accepting or not depending on a language of (accepting) infinite words W of control states. Parity and weak parity ω-ATM are defined accordingly.

Lemma

A languages $L \subseteq \Sigma^*$ is Σ^0_{n+1} (resp. Π^0_{n+1}) if and only if it is definable by means of an ω-ATM with weak parity condition of range $[1, n+1]$ (resp. $[0, n]$).

Definition (ω-ATM)

An ω-Alternating Turing Machine is an Alternating Turing Machine (ATM) possibly with infinite runs that are said accepting or not depending on a language of (accepting) infinite words W of control states. Parity and weak parity ω-ATM are defined accordingly.

Lemma

A languages $L \subseteq \Sigma^*$ is Σ^0_{n+1} (resp. Π^0_{n+1}) if and only if it is definable by means of an ω-ATM with weak parity condition of range $[1, n+1]$ (resp. $[0, n]$).

Theorem

Computing winning strategies in two processes distributed game with weak parity conditions is:

1. Π^0_n-*complete with range* $[0, n-1]$,
2. Σ^0_n-*complete with range* $[1, n]$.

Proof.

(lower bound) Shift from solitaire domino games to two player, henceforth alternating, domino games with weak parity condition. □

Theorem

Computing winning strategies in two processes distributed game with weak parity conditions is:

1. Π_n^0-*complete with range* $[0, n-1]$,
2. Σ_n^0-*complete with range* $[1, n]$.

Proof.

(lower bound) Shift from solitaire domino games to two player, henceforth alternating, domino games with weak parity condition. □

Theorem

Computing winning strategies in two processes distributed game with weak parity conditions is:

1. Π^0_n-*complete with range* $[0, n-1]$,
2. Σ^0_n-*complete with range* $[1, n]$.

Proof.

(lower bound) Shift from solitaire domino games to two player, henceforth alternating, domino games with weak parity condition. □

Theorem

Computing winning strategies in two processes distributed game with parity condition is Σ_1^1-complete.

Proof.

(lower bound) Infinite tiling with Büchi condition (parity cond. with range [0,1]) are Σ_1^1-complete hence two process distributed games with Büchi conditions. □

Theorem

Computing winning strategies in two processes distributed game with parity condition is Σ^1_1-complete.

Proof.

(lower bound) Infinite tiling with Büchi condition (parity cond. with range [0,1]) are Σ^1_1-complete hence two process distributed games with Büchi conditions. □

- No applications !
- But a better understanding of undecidability sources !
- And an (ignored ?) interesting relationship between weak parity conditions and the arithmetical hierarchy.

- No applications !
- But a better understanding of undecidability sources !
- And an (ignored ?) interesting relationship between weak parity conditions and the arithmetical hierarchy.

- No applications !
- But a better understanding of undecidability sources !
- And an (ignored ?) interesting relationship between weak parity conditions and the arithmetical hierarchy.

- No applications !
- But a better understanding of undecidability sources !
- And an (ignored ?) interesting relationship between weak parity conditions and the arithmetical hierarchy.