Games in logic

Games are used to capture "dynamics" of formulas. They offer an understanding of formula constructors in "operational" way. The three settings presented here focus on different constructors and in consequence use very different techniques.

- Ehrenfeucht-Fraïssé games focus on the behaviour of quantifiers. This leads to a notion of type and to the compositional method.
- Party games are used to understand operators defined by fixpoints. This leads to theory of automata on infinite words or trees.
- Game semantics is used to understand dynamics of propositional constructs. This leads to accurate models of proofs and of programming languages.

Question: What are the connections between these three settings?

Plan

Games in logic

Part I

Ehrenfeucht-Fraïssé games and the composition method

Ehrenfeucht-Fraïssé games focus on the behaviour of quantifiers. This leads to a notion of type and to the compositional method. In consequence we study how to cut structures so that from the theory of parts one can obtain the theory of the whole. The other application is to obtain normal forms of formulas.

+ Monadic second order logic
+ E-F games and the composition theorem for sums (Shelah’s way).
+ CTL=Chain Logic.

[Thomas, “Ehrenfeucht Games, the Composition Method, and the Monadic Theory of Ordinal Words.” Structures in Logic and Computer Science, LNCS 1261]

MSOL and Ehrenfeucht-Fraïssé games

Monadic second order logic

- Instead of quantification over elements we have quantification over sets.
  \( \exists X \forall y(X) \quad \forall X \forall y(X) \)
- We have the inclusion predicate
  \( X \subseteq Y \)
- Standard predicates can be "lifted" to sets:
  \( \exists x \in \{ X, Y \}, \quad X \subseteq Y, \quad X \subseteq P \)

Ehrenfeucht-Fraïssé games for MSOL

- We have two structures \( (A, P) \) and \( (B, Q) \) with some distinguished sets of elements.
- We also have \( \pi = (k_1, \ldots, k_n) \), a vector of positive natural numbers.
- If \( \pi \) is empty then Duplicator wins iff the two structures satisfy the same predicates with respect to \( P \) and \( Q \). Otherwise Spoiler wins.
- If \( \pi = \pi + m \) then Spoiler chooses one of the structures, say \( A \), and \( m \) sets in this structure \( P_1, \ldots, P_m \). Duplicator replies by choosing \( Q_1, \ldots, Q_m \), in \( B \) in the same order and the \( \pi \)-game is played on \( (A, P_1, \ldots, P_m) \) and \( (B, Q_1, \ldots, Q_m) \).

Composition theorems

Sum

The \( \pi \)-type of \( A + B \) is determined by (and can be computed from) the \( \pi \)-types of \( A \) and \( B \).

Ordered sum

The \( \pi \)-type of \( \sum^
\pi A_i \) is determined by (and can be computed from) the \( \pi \)-types of \( A_i \).

Important property: \( (A, P) -\pi\equiv (B, Q) \iff T_\pi(A, P) -\pi\equiv T_\pi(B, Q) \).
Deciding logics using compositional theorems

- A $\tau$-type of a structure tells what $\tau$-formulas are true in the structure.
- First, we will calculate possible types of finite sequences.
- Then, we calculate the types for infinite sequences.

### Computing the theory of finite sequences

#### The theory of finite sequences

- Consider structures $A_n = ((1, \ldots, n), \leq)$.
- Let $\mathcal{T}_P^{\tau_k}(A_n) = \{ \tau^k(A_n) : \tau \in \mathcal{L}(A_n) \}$ (these are all possible $\tau$-types of finite sequences).

#### Computing $\mathcal{T}_P^{\tau_k}(\text{Fin}(m))$

- $\mathcal{T}_P^{\tau_k}(A_n) = T_{\tau_k(n)} + T_{\tau_k}(A_{n+1})$
- $T_{\tau_k(n+1)} = T_{\tau_k(n)} + T_{\tau_k}(A_{n+1})$

We take $\mathcal{T}_P^{\tau_k}(\text{Fin}(m)) = \bigcup_{n=0}^{m} T_{\tau_k(n)}$.

**Remark**

$\mathcal{T}_P^{\tau_k}(\text{Fin}(m))$ gives us all possible $\tau$-types of $((1, \ldots, n), \leq, A_1, \ldots, A_n)$ for $n \in \mathbb{N}$.

### Computing the theory of infinite sequences

#### The base step: $\mathcal{T}_P^{\tau_k}(\omega)$

We compute by hand the theory $\mathcal{T}_P^{\tau_k}(\omega)$ that is:

- $\mathcal{T}_P^{\tau_k}(\omega) = \mathcal{T}_P^{\tau_k}(\omega \cdot 1)$

#### Observation

- If we can compute $\mathcal{T}_P^{\tau_k}(\omega)$ then we can compute $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q})$ where only one $\bar{Q}_i = \omega$ and the rest is $\emptyset$.

#### Induction step for $\mathcal{T}_P^{\tau_k}(\omega)$

- $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q}) = \mathcal{T}_P^{\tau_k}(\omega, \bar{Q}_i)$ for $i \leq n$. Each $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q})$ can be presented as $\tau_i + \sum \sigma_i$ for $\sigma_i \in \mathcal{T}_P^{\tau_k}(m)$.

#### Composing $\sum \sigma_i$ reduces to computing $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q}_i)$ where only one $\bar{Q}_i$ is $\omega$ and the rest is $\emptyset$.

### Computing the theory of $\langle \omega, \leq \rangle$

#### The base step: $\mathcal{T}_P^{\tau_k}(\omega)$

We compute by hand the theory $\mathcal{T}_P^{\tau_k}(\omega)$ that is:

- $\mathcal{T}_P^{\tau_k}(\omega) = \mathcal{T}_P^{\tau_k}(\omega \cdot 1)$

#### Observation

- If we can compute $\mathcal{T}_P^{\tau_k}(\omega)$ then we can compute $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q})$ where only one $\bar{Q}_i = \omega$ and the rest is $\emptyset$.

### Induction step for $\mathcal{T}_P^{\tau_k}(\omega)$

- $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q}) = \mathcal{T}_P^{\tau_k}(\omega, \bar{Q}_i)$ for $i \leq n$. Each $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q})$ can be presented as $\tau_i + \sum \sigma_i$ for $\sigma_i \in \mathcal{T}_P^{\tau_k}(m)$.

#### Composing $\sum \sigma_i$ reduces to computing $\mathcal{T}_P^{\tau_k}(\omega, \bar{Q}_i)$ where only one $\bar{Q}_i$ is $\omega$ and the rest is $\emptyset$.

### Understanding logics on trees

#### A composition theorem for trees

- We use first-order logic over predicates $x \leq y$, $y \in \ell(P)$, and $\tau(y, x)$.
- We will not need vectorial ranks. So we write $\mathcal{T}_P^{\tau}(A)$ for the $\tau$-type of $A$, where $A \in \mathbb{N}$.

#### From trees to paths: $\exp^\tau(t, u)$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$u$</th>
<th>$\exp^\tau(t, u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$w$</td>
<td>$\mathcal{T}_P^{\tau_k}(\ell(P) \cdot v_i)$</td>
</tr>
<tr>
<td>$n$</td>
<td>$w$</td>
<td>$\mathcal{T}_P^{\tau_k}(\ell(P) \cdot v_j)$</td>
</tr>
<tr>
<td>$m$</td>
<td>$u$</td>
<td>$\mathcal{T}_P^{\tau_k}(\ell(P) \cdot v_j)$</td>
</tr>
<tr>
<td>$m$</td>
<td>$u$</td>
<td>$\mathcal{T}_P^{\tau_k}(\ell(P) \cdot v_j)$</td>
</tr>
</tbody>
</table>

#### Theorem

The $\tau$-type of a finite binary tree $t$ is determined by the set $\{ \exp^\tau(t, u) : u \in \tau \}$.
Verification as a game

Verification (Model Checking)

Given a transition system $M$ and a property $\psi$, check if $M \models \psi$.

Reformulation

Construct a game $G(M, \psi)$ of two players: Adam and Eve.

Fix the rules in such a way that Eve wins from the initial position of $G(M, \psi)$ iff $M \models \psi$.

Different Games for Different Properties

$M, s \models \psi$

$G(M, s, \alpha)$

Modal Logic

Finite Duration Games

Reachability

Termination

Safety

Non-Termination

Example

Game: $G = \langle V, E, \lambda : V \to C, \text{Acc} \subseteq C^\omega \rangle$

Definition (Game $G = \langle V, E, \lambda : V \to C, \text{Acc} \subseteq C^\omega \rangle$)

Definition (Winning a Play)

Eve wins a play in $\ldots$ if the sequence is in $\text{Acc}$.

Definition (Winning Position)

A strategy for Eve is $\sigma_E : V^* \times V_E \to V$. A strategy is winning from a given position iff all the plays starting in this position and respecting the strategy are winning. A position is winning if there is a winning strategy from it.
What kind of winning conditions

The parity condition

Part IIb

Syntactic properties

Games for the μ-calculus

Game rules

Example: Reachability

Example: Reachability

Example: Reachability

Winning conditions

The μ-calculus

Games for the μ-calculus

Syntactic properties

Games for the μ-calculus

Game rules

Example: Reachability

Example: Reachability

Example: Reachability
Different ways of winning and playing games.

**Example: Reachability**

Reachability: \( \{P \} P = \nu X. P \lor \{X \} \)

- \( \alpha \equiv \nu X. P \lor \{X \} \)
- \( P \lor \{\} \)
- \( P \lor \{\} \)

Eve wins if the game ends in \( P \). \( \mu X. P \lor \{X\} \) is represented.

**Example: Reachability**

Reachability: \( \{P \} P = \nu X. P \lor \{X \} \)

- \( \alpha \equiv \nu X. P \lor \{X\} \)
- \( P \lor \{\} \)
- \( P \lor \{\} \)

Eve wins if the game ends in \( P \). \( \mu X. P \lor \{X\} \) is represented.

**Example: Reachability**

Reachability: \( \{P \} P = \nu X. P \lor \{X\} \)

- \( \alpha \equiv \nu X. P \lor \{X\} \)
- \( P \lor \{\} \)
- \( P \lor \{\} \)

Eve wins if the game ends in \( P \). \( \mu X. P \lor \{X\} \) is represented.

**Example: Reachability**

Reachability: \( \{P \} P = \nu X. P \lor \{X\} \)

- \( \alpha \equiv \nu X. P \lor \{X\} \)
- \( P \lor \{\} \)
- \( P \lor \{\} \)

Eve wins if the game ends in \( P \). \( \mu X. P \lor \{X\} \) is represented.

**Defining winning conditions**

- \( \nu X \alpha \)
- \( \nu X \alpha \)
- \( \nu X \alpha \)

\( \alpha \) wins if the smallest priority appearing infinitely often is even.

**Example**

\( \nu X. Y. P \lor \{X\} \lor \{X\} \)

\( \nu X. Y. P \lor \{X\} \lor \{X\} \)

**Results on payoff games**

**Theorem (Herrmann & Mycielski)**

Every vertex of a mean payoff game has a value. Moreover the two players have positional optimal strategies.

**Theorem (Zwick & Paterson)**

For every finite discounted payoff game the value exists in every vertex and is given as a unique solution of the set of equations:

\[ v_t = (1 - \delta) v_{t+1} + \max_{a \in A} \min_{x \in X} c_{t}(a,x) \]

and

\[ v_t = (1 - \delta) v_{t+1} + \min_{a \in A} \max_{x \in X} c_{t}(a,x) \]

There are optimal positional strategies.

**Theorem (Zwick & Paterson)**

When \( \delta \to 1 \) then \( V^{\text{DP}}(v) \to V^{\text{DP}}(v) \).
**Perfect information stochastic games**

**Definition**
Apart from positions for Eve and Adam there are randomized positions where a successor is chosen according to a probability distribution.

**Example**

Eve wins with the probability 2/3 and Adam with the probability 1/3.

**Theorem (de Alfaro, Majumdar, Chatterjee & Jurzyczki & Henzinger, Zielonka)**
In a finite game each state has a value and each player has a positional pure and optimal strategy.

**Part III**

**Game semantics**
Game semantics is used to understand dynamics of propositional constructs. It extracts computational content from proofs, abstracting from irrelevant detail. This leads to accurate models of proofs and programming languages. As for E-F types, it is not the winning the matters, but the ways to play.

- Proofs as games
- Game semantics: an example
- Brief summary of the results, and (potential) applications

**Satisfiability in propositional logic**

**We have examined \( a \vdash a \) now we look at \( a \vdash \beta \)**

**Game rules**

\[
\begin{align*}
\alpha \lor \beta & \vdash a \lor \alpha \\
\alpha \lor \beta & \vdash a \lor \beta \\
\alpha \lor \beta & \vdash a \lor \beta \\
\alpha \lor \beta & \vdash a \lor \beta \\
\alpha \lor \beta & \vdash a \lor \beta \\
\end{align*}
\]

**What one can prove with these rules?**
Not much. These rules characterize \( \lor \) and \( \land \) in the free lattice.

**Proofs as strategies**

**Strategies instead of proofs**

\[
\begin{align*}
\vdash a \rightarrow (a \rightarrow a) \\
\vdash a \rightarrow (a \rightarrow a) \\
\vdash a \rightarrow (a \rightarrow a) \\
\vdash a \rightarrow (a \rightarrow a) \\
\vdash a \rightarrow (a \rightarrow a) \\
\end{align*}
\]

**Proofs as programs**

\[
\begin{align*}
\lambda x. y . z : N \rightarrow N \rightarrow N \\
\lambda x. y . z : N \rightarrow N \rightarrow N \\
\end{align*}
\]
Program semantics as a strategy
\[ f : N \to N \vdash \text{if } W f (5) = 6 \begin{cases} 7 & \text{then} \\ 0 & \text{else} \end{cases} \]

Remarks
- Different strategies correspond to different programs.
- The semantics is not about provability.
- Semantics of a program is, roughly, a set of words: all plays permitted by the strategy.

Idealized Algol

Types
- \( b \colon \text{com} \mid \text{exp} \mid \text{var} \quad \tau = \{ b \mid \tau \to \tau \}

Rules
- \( \Gamma \vdash \text{skip} : \text{com} \mid \text{exp} \quad \tau \in \{0, \ldots, \infty\} \quad \Gamma, \tau \vdash \tau \to \tau \)
- \( \Gamma, x : \tau \vdash M : \tau' \quad \Gamma, \tau \vdash \tau' \to \tau' \quad \Gamma, M : \tau \vdash \tau' \)
- \( \Gamma \vdash \text{while} \ M \ \text{do} \ N : \text{com} \mid \text{exp} \quad \Gamma, x : \tau \vdash M : \tau \)

Iteration and recursion
- \( \Gamma, x : \tau \vdash \text{while} \ M \ \text{do} \ N : \text{com} \mid \text{exp} \quad \Gamma, x : \tau \vdash M : \tau \)

Game semantics

Observational equivalence
Two terms are observationally equivalent, \( M \approx N \), if they behave the same in all contexts
- \( C[M] \parallel \text{skip} \) if \( C[N] \parallel \text{skip} \)
- Quantification over all program contexts \( C \vdash M \) makes this notion hard to work with.
- The notion is very compelling as it captures well semantical indistinguishability.

Theorem (Abramsky & McCusker)
The game semantics of \( L_A \) is fully abstract, which means that for any \( M \) and \( N \):
- \( [M] = [N] \) if and only if \( M \approx N \).

Reasoning about programs using game semantics

More examples

Calling a function twice
\[ f : N \to N \vdash f(5) : N \]

Nested calls
\[ f : N \to f(5) : N \]

Reasoning about program equivalence
- Construct the strategy representing a given program (typically using finite approximations for data domains).
- Sometimes this strategy can be represented by a finite automaton, or by a deterministic pushdown automation.
- If so, we can check program equivalence by comparing languages for the two automata.

Results
- For Idealized Algol we have quite good understanding for which subclasses the equivalence problem is decidable.

Conclusions
- In E-F games, the semantics of a formula in a structure is represented by a type: a tree describing all potential extensions.
- In parity games, the semantics of a fixpoint formula is represented as an infinite unfolding of a formula. Winning conditions on infinite plays are needed to handle fixpoints of different types.
- In game semantics, formula, or program, is represented by a set of plays it admits. This set can be sometimes regular or context-free.

Questions
- Are there connections between these settings?
- Are there nontrivial results that can be transferred from one setting to another?