ALGORITHMIC GAMES FOR FULL GROUND REFERENCES

Andrzej Murawski
University of Warwick

Nikos Tzevelekos
Queen Mary
University of London

CONTEXTUAL EQUIVALENCE

$$M_1 \cong M_2$$

EASIER GRAND CHALLENGE?

Program equivalence can be thought of as a grand challenge in its own right, but there are reasons to believe that it is a lower hanging fruit.

Godlin & Strichman

λ ref

FULL GROUND REFERENCES

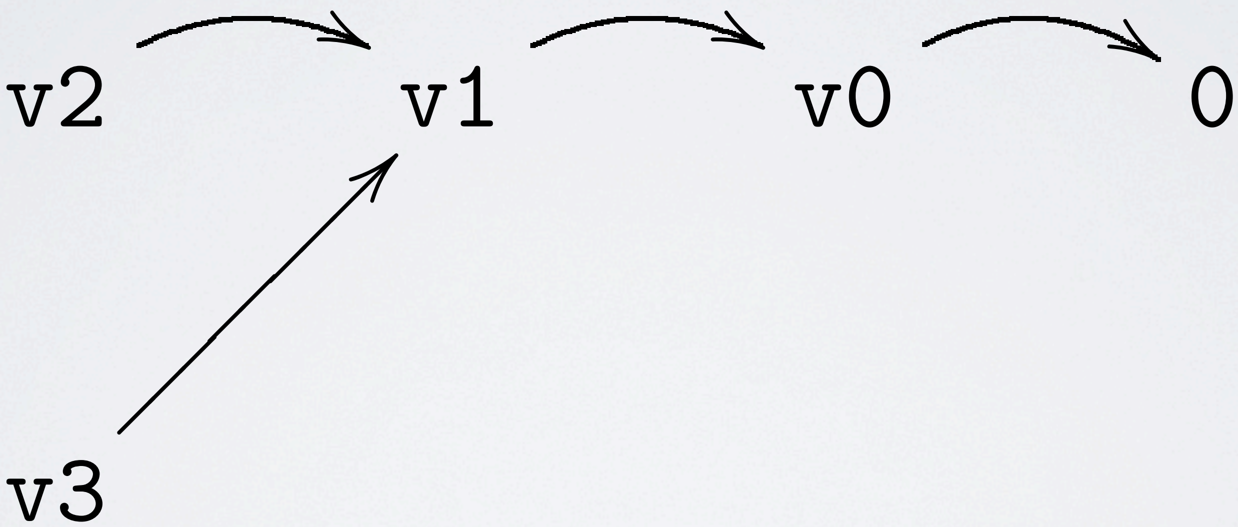
```
# let v0=ref(0);;
val v0 : int ref = {contents = 0}

# let v1=ref(v0);;
val v1 : int ref ref = {contents = {contents = 0}}

# let v2=ref(v1);;
val v2 : int ref ref ref = {contents = {contents = {contents = 0}}}

# let v3=ref(v1);;
val v3 : int ref ref ref = {contents = {contents = {contents = 0}}}
```

POINTERS



- well-founded
- no pointer arithmetic

TYPES

$$\theta \quad ::= \quad \mathsf{unit} \quad | \quad \mathsf{int} \quad | \quad \mathsf{ref}^k(\mathsf{int}) \quad | \quad \theta \to \theta$$

TERMS

$()$ 0 1 $\cdots$ max

$\text{case}(M)[N_0, \cdots, N_{max}]$ $\text{while } M \text{ do } N$

x $\lambda x^\theta.M$ $M\,N$

$\text{ref}(M)$ $!M$ $M := N$

CONTEXTUAL EQUIVALENCE

$$\Gamma \vdash M : \theta$$

$\Gamma \vdash M_1 : \theta$ and $\Gamma \vdash M_2 : \theta$ are ***contextually equivalent*** provided, for any context $C[-]$,

$$C[M_1] \Downarrow \text{ if and only if } C[M_2] \Downarrow .$$

We then write $\Gamma \vdash M_1 \cong M_2 : \theta$.

EXAMPLE

$$p : \mathsf{ref}(\mathsf{int}) \to \mathsf{unit} \quad \vdash \quad \begin{array}{l} \mathsf{let}\ x = \mathsf{ref}(0)\ \mathsf{in} \\ \mathsf{let}\ y = \mathsf{ref}(x)\ \mathsf{in} \\ p(x); \\ \mathsf{if}\ (!y = x)\ \mathsf{then}\ \mathrm{diverge}_{\mathsf{unit}}\ \mathsf{else}\ () \end{array}$$

$$\cong$$

$$\mathrm{diverge}_{\mathsf{unit}} \qquad : \mathsf{unit}$$

$$\mathsf{ref}(\mathsf{int}) \to \mathsf{unit} \quad \vdash \quad \mathsf{unit}$$

QUESTION

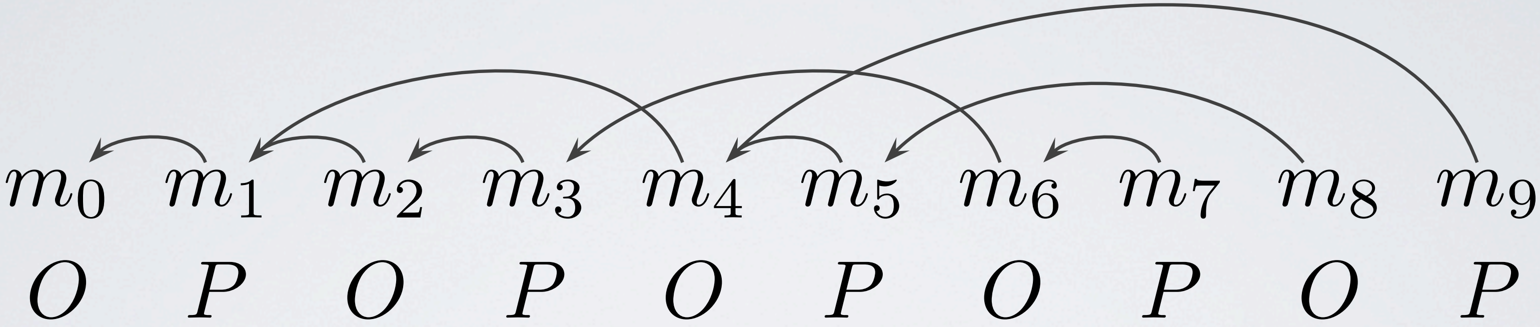
For which types $\theta_1, \cdots, \theta_n, \theta$ is it possible to decide contextual equivalence between terms of the shape

$$x_1 : \theta_1, \cdots, x_n : \theta_n \vdash M : \theta?$$

TECHNIQUES

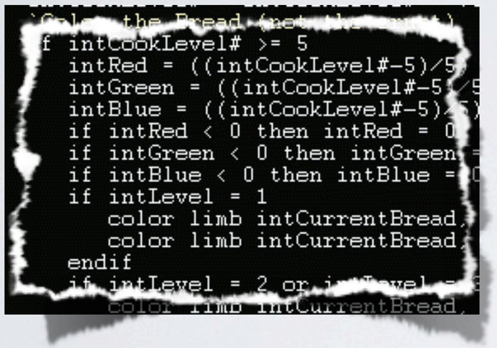
- logical relations
- environmental bisimulation
- trace semantics
- game semantics

GAME SEMANTICS

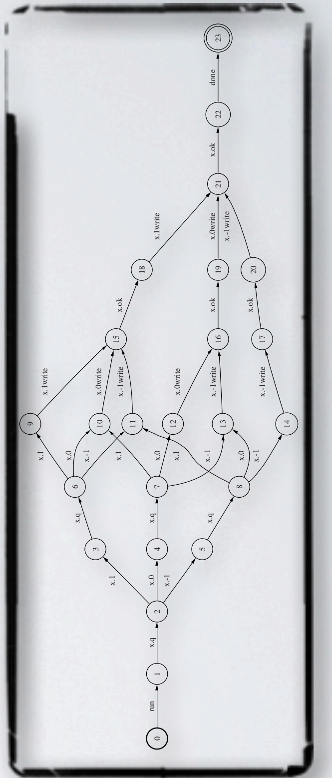


- Two players: **O** (System) and **P** (Program)
- Programs are interpreted as strategies for **P**.
- **Compositional** interpretation
- **Automata-theoretic** representations

GAME SEMANTICS



strategy



M_1, M_2 contextually equivalent $\iff$ $[\![M_1]\!] = [\![M_2]\!]$ $\iff$ $\mathcal{A}_{M_1} \approx \mathcal{A}_{M_2}$

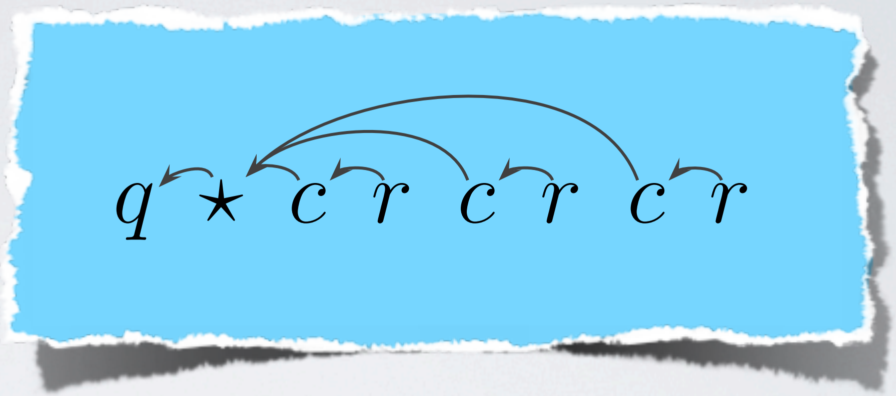
GAME MODELS

- J. Laird. *A game semantics of names and pointers.* Ann. Pure Appl. Logic 151(2-3): 151-169 (2008)
- A. S. Murawski and N. Tzevelekos. *Game semantics for good general references.* LICS 2011: 75-84

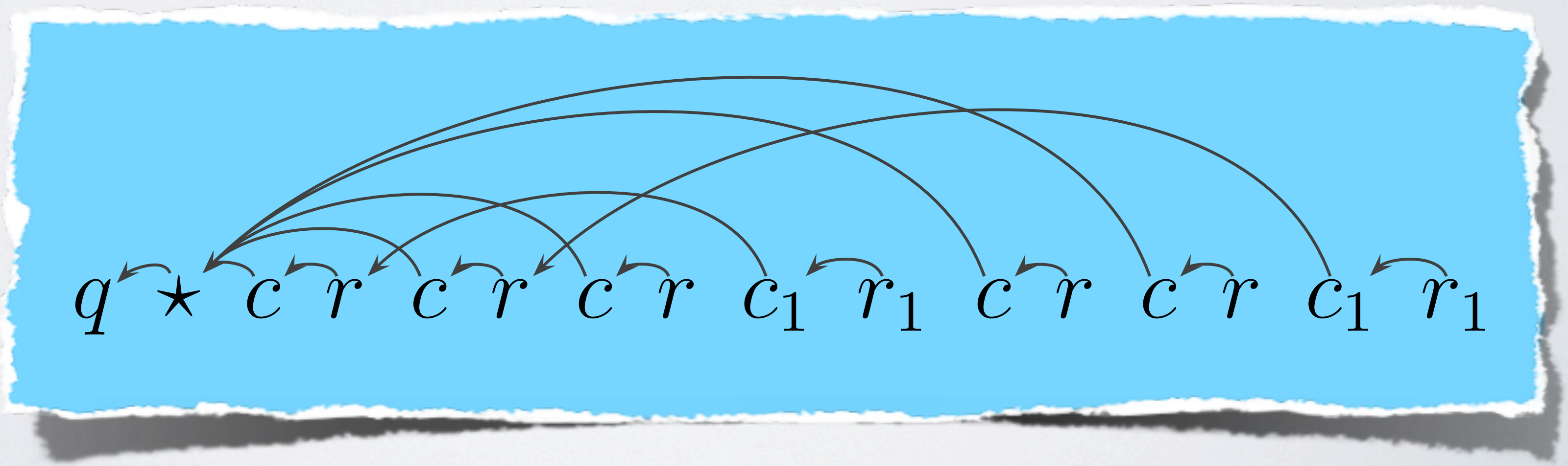
METHODOLOGY

- **Investigate the shape of plays for given types.**
- **Try to find decidable classes of machine models that can represent the corresponding plays.**
- **If they are complicated enough, try to use them to support a simulation of a Turing-complete formalism.**

$\vdash \mathsf{unit} \to \mathsf{unit}$



$\vdash \mathsf{unit} \to \mathsf{unit} \to \mathsf{unit}$



FULL CLASSIFICATION

$$\cdots, \theta_L, \cdots \vdash \theta_R$$

θ_R	decidability
unit	☺
unit → unit	☺
(unit → unit) → unit	☺
((unit → unit) → unit) → unit	☹
unit → unit → unit	☹

LHS

$$\theta_L \quad \equiv \quad \theta_R \rightarrow \ldots \rightarrow \theta_R \rightarrow \mathsf{unit}$$

NOMINAL GAMES

- $\vdash$ let $n =$ ref(0) in $\lambda x^{\mathsf{unit}}.n$

$$q \ \star \ c \ n^{(n,0)} \ c^{(n,5)} \ n^{(n,5)} \ c^{(n,12)} \ n^{(n,12)} \ldots$$

- $\vdash \lambda x^{\mathsf{unit}}.ref(0)$: unit $\rightarrow$ ref

$$q \ \star \ c \ n_1^{(n_1,0)} \ c^{(n_1,5)} \ n_2^{(n_1,5),(n_2,0)} \ c^{(n_1,12),(n_2,7)} \ n_3^{(n_1,12),(n_2,7),(n_3,0)} \ldots$$

UNBOUNDED GROWTH

$$q \star c\, n_1^{(n_1,0)}\; c^{(n_1,5)}\; n_2^{(n_1,5),(n_2,0)}\; c^{(n_1,12),(n_2,7)}\; n_3^{(n_1,12),(n_2,7),(n_3,0)} \cdots$$

can be faithfully represented by

$$q \star c\, n_1^{(n_1,0)}\; c\, n_2^{(n_2,0)}\; c\, n_3^{(n_3,0)}$$

PUSHDOWN AUTOMATA [CK98]

locally/globally fresh

- Q *is a finite set of states.*
- $s_0 \in Q$ *is the initial state.*
- $\boldsymbol{u} = u_1 u_2 \cdots u_r \in \Sigma^{r\neq}$, *is the* initial assignment *to the* r *registers of* $\mathscr{A}$.
- $\rho : Q \to \{1, 2, \ldots, r\}$ *is a partial function from* Q *to* $\{1, 2, \ldots, r\}$ *called the* reassignment. *Intuitively, if* $\mathscr{A}$ *is in state* q, *and* $\rho(q)$ *is defined, then* $\mathscr{A}$ *can* non-deterministically *replace the content of the* $\rho(q)$*th register with a new symbol of* Σ *not appearing in any other register. Note that, unlike in [5], we allow* $\mathscr{A}$ *to guess the replacement. This is essential, because grammars can guess symbols they generate.*
- μ *is a mapping from* $Q \times (\{1, 2, \ldots, r\} \cup \{\varepsilon\}) \times \{1, 2, \ldots, r\}$ *to finite subsets of* $Q \times \{1, 2, \ldots, r\}^*$ *called the* transition function. *Intuitively, if* $(p, j_1 j_2 \cdots j_n) \in \mu(q, k, i)$, $n \geq 0$, *then (after reassigning the* $\rho(q)$*th register)* $\mathscr{A}$, *whenever it is in the state* q, *with content of the* i*th register at the top of the stack, and the input symbol equal to the content of the* k*th register, can replace the top symbol on the stack with the content of* j_1*th,* j_2*th, ...,* j_n*th registers (in this order, read top-down), enter the state* p, *and pass to the next input symbol (possibly* ε*). Similarly, if* $(p, j_1 j_2 \cdots j_n) \in \mu(q, \varepsilon, i)$, *then A, whenever it is in the state* q, *with content of the* i*th register at the top of the stack, can replace the top symbol on the stack with the content of* j_1*th,* j_2*th, ...,* j_n*th registers, enter the state* p *(without reading the input symbol), and pass to the next input symbol (possibly* ε*).*

EQUIVALENCE TESTING

- **Not a direct language equivalence test.**
- **Store matching needs to take place.**
- **Local/global freshness clashes have to be handled.**
- **Emptiness testing.**

SUMMARY

- **Programming with references**
- **Contextual equivalence**
- **Nominal game semantics**
- **Automata over infinite alphabets**