ALGORITHMIC GAME SEMANTICS

Andrzej S. Murawski

UNIVERSITY OF LEICESTER

GAME SEMANTICS

```
et rec add_identifier id id_type
 match env with
   [ ] -> [ ( id,[id_type]) ]
 | (top_id, ty) :: env_tl  ->
     if top_id = id then (top_ic
     else (top_id, ty) :: add_ic

.et rec rem_identifier id env =
 match env with
   [ ] -> raise Undeclared_ident
 | ( top_id, ty) :: env_tl  ->
     if top_id = id then (if ty=
nv_tl)
     else (top_id, ty) :: rem_id
```

FULL ABSTRACTION

M and N are ***contextually equivalent*** ($M \cong N$) if they can be used interchangeably in any context (without affecting the computational outcome).

$$\forall\, \mathbb{C}[-]. \qquad \mathbb{C}[M] \Downarrow \quad \Longleftrightarrow \quad \mathbb{C}[N] \Downarrow$$

$$[\![M]\!] = [\![N]\!] \iff M \cong N$$

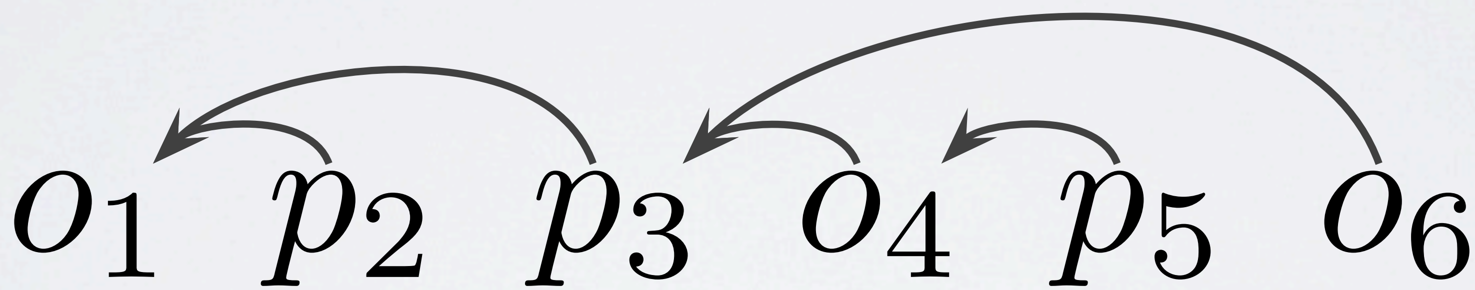
GAMES FOR TYPES

- **Who plays?**

O	**P**
Opponent	Proponent
$\mathbb{C}[-]$	M

JUSTIFIED SEQUENCES

- **How do they play?**

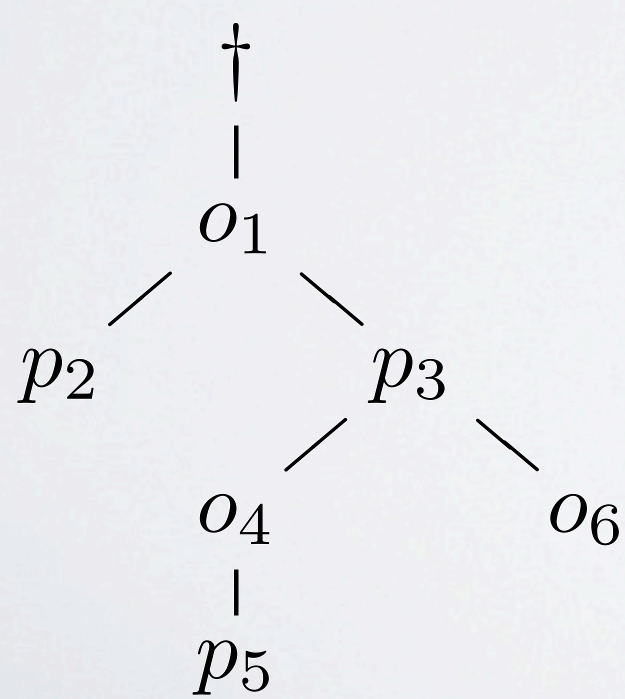


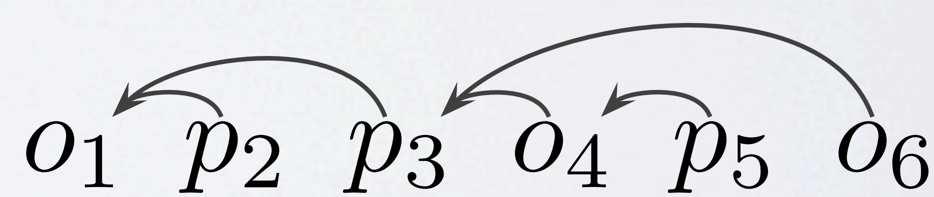
O begins. Subsequent moves must be justified by earlier moves made by the opposite player .

GAMES PLAYED IN ARENAS

An ***arena*** A is specified by a structure $\langle M_A, \lambda_A, \vdash_A \rangle$.

- M_A is a set of *moves*.
- $\lambda_A : M_A \to \{O, P\} \times \{Q, A\}$ is a *labelling* function.
- $\vdash_A$ is an *enabling* relation between $\{\dagger\} + M_A$ and M_A.
  - If $\dagger \vdash m$ then $\lambda_A(m) = O$ and $n \vdash_A m$ implies $n = \dagger$.
  - If $m \vdash m'$ then $\lambda_A(m) \neq \lambda_A(m')$.





PLAYS

A ***justified sequence*** over arena A is a sequence of moves from M_A together with an associated sequence of pointers satisfying the following conditions.

- The first move is enabled by $\star$ and has no outgoing pointer.
- Any other move m must have a pointer to an earlier move n such that $n \vdash_A m$.

N.B. Papers on game semantics use variations on the concept of a justified sequence to suit the programming paradigm being modelled.

A ***play*** is a justified sequence that additionally satisfies ...
We shall write P_A for the set of plays over arena A.

SOME EXAMPLES

- Sequential computation: **alternation**
- Absence of control effects: **well-bracketing**

$$\cdots \quad q \;\; q_1 \;\; a_1 \quad \cdots \quad q_n \;\; a_n \;\; a \quad \cdots$$

- First-order store only: **visibility**

$$o_1 \;\; p_1 \quad \cdots \quad o_2 \;\; p_2 \quad \cdots \quad o_3 \quad p_3 \quad \cdots \quad o_4$$

In his next move P cannot use $\cdots$ for justification.

HISTORY

All the conditions were already present in

J. M. E. Hyland, C.-H. Luke Ong: On Full Abstraction for PCF: I, II, and III. Inf. Comput. 163(2): 285-408 (2000)

But it took a few years to match them with other computational paradigms.

Samson Abramsky, Kohei Honda, Guy McCusker: A Fully Abstract Game Semantics for General References. LICS 1998: 334-344

James Laird: Full Abstraction for Functional Languages with Control. LICS 1997: 58-67

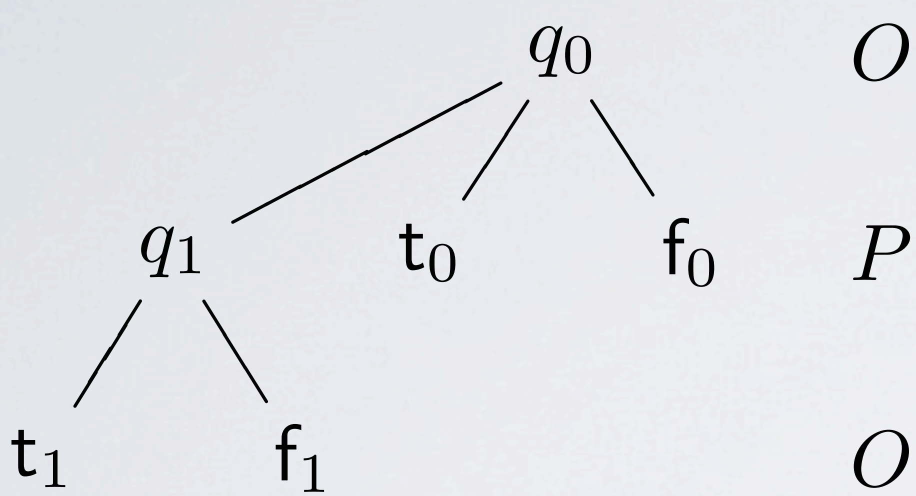
Samson Abramsky, Guy McCusker: Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. Electr. Notes Theor. Comput. Sci. 3: 2-14 (1996)

REASONING WITH GAMES

- Plays have operational flavour.
- The course of play is often described through metaphores.
- This account has not been formalized yet.
- Operational game semantics: marriage of games and traces

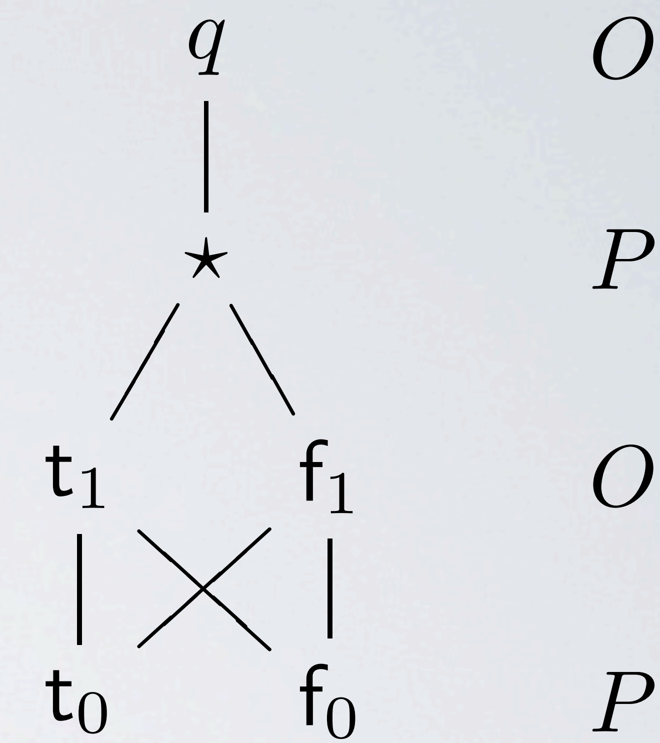
$\vdash \mathsf{bool} \to \mathsf{bool}$

CBN



Questions	q_0, q_1
Answers	$\mathsf{t}_0, \mathsf{f}_0, \mathsf{t}_1, \mathsf{f}_1$

CBV



Questions	$q, \mathsf{t}_1, \mathsf{f}_1$
Answers	$\star, \mathsf{t}_0, \mathsf{f}_0$

Plays

$$q_0 \ (q_1 \ b_1)^* \ b_0$$

$$q \ \star \ (b_1 \ b_0)^*$$

STRATEGIES

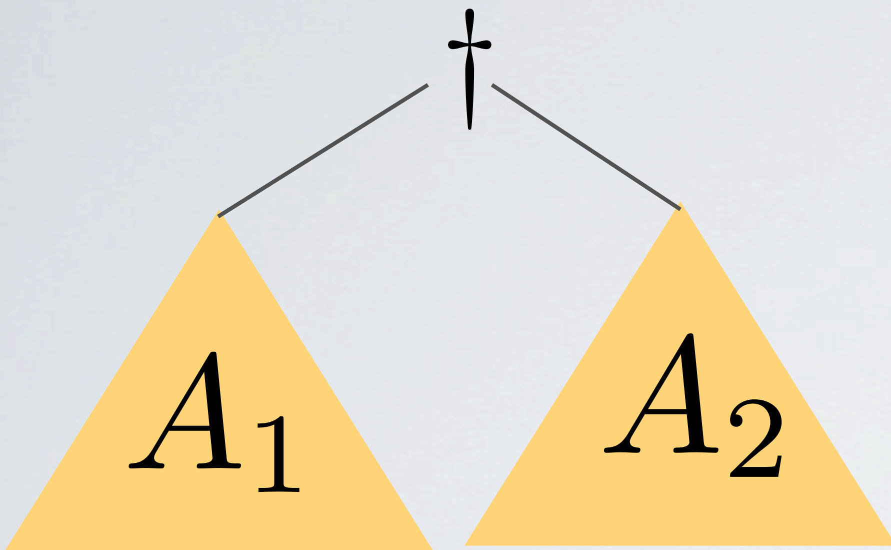
- Types are interpreted by games.
- Terms are interpreted by strategies.

A ***strategy*** σ in arena A is a prefix-closed set of *plays* over A such that

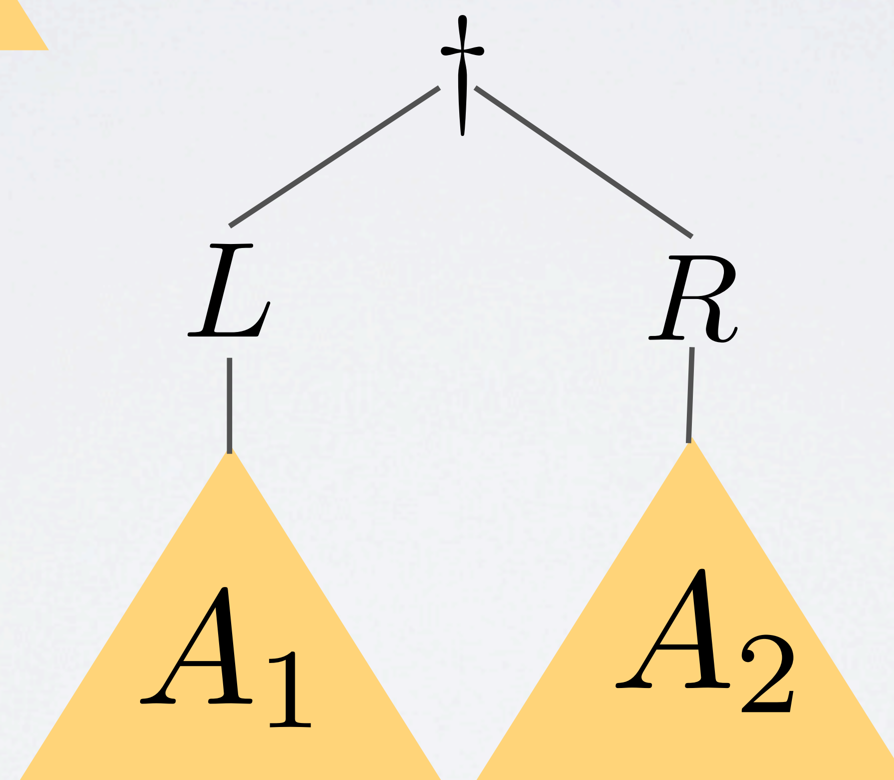
$$s \in \sigma \text{ and } s\,o \in P_A \text{ implies } so \in \sigma\,.$$

Games and strategies are treated as first-class mathematical objects.

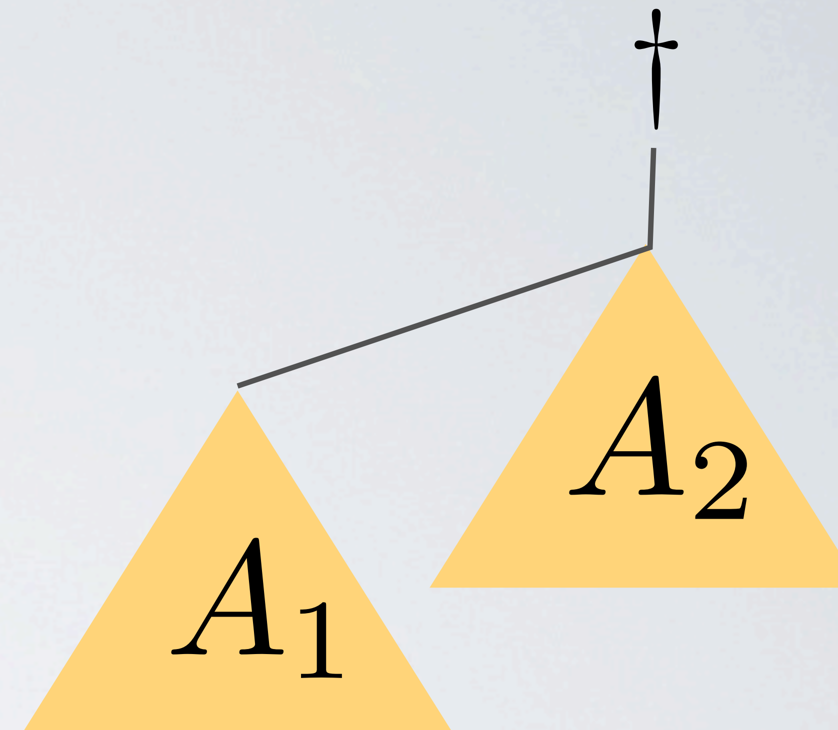
GAME CONSTRUCTORS



$A_1 \times A_2$



$A_1 + A_2$

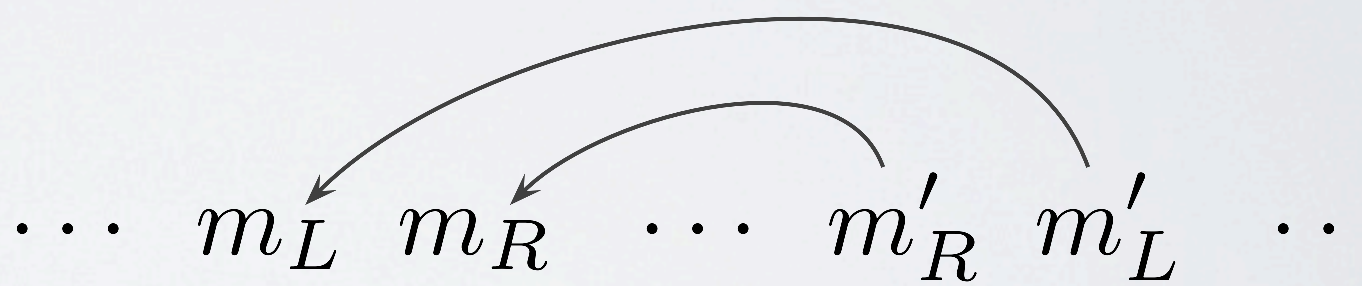


$A_1 \Rightarrow A_2$

IDENTITY STRATEGY

$$A \quad \Rightarrow \quad A$$

	$\vdots$	
O	m_L	
P		m_R
	$\vdots$	
O		m'_R
P	m'_L	
	$\vdots$	



COMPOSITION

Given $\sigma : A_1 \Rightarrow A_2$ and $\tau : A_2 \Rightarrow A_3$
one can define $\sigma;\tau : A_1 \Rightarrow A_3$.

- Moves in A_2 have a double identity.
- We can exploit the duality to play σ and τ against each other in A_2.
- Following the exchange between σ and τ we can hide the interaction in A_2 to obtain a play in $A_1 \Rightarrow A_3$.

COMPOSITION

A_1	$\overset{\sigma}{\Rightarrow}$	A_2	$\overset{\tau}{\Rightarrow}$	A_3
				o
				p
				o
		o/p		
p				
o				
		p/o		
		o/p		
p				
o				
		p/o		
		o/p		
		p/o		
				p

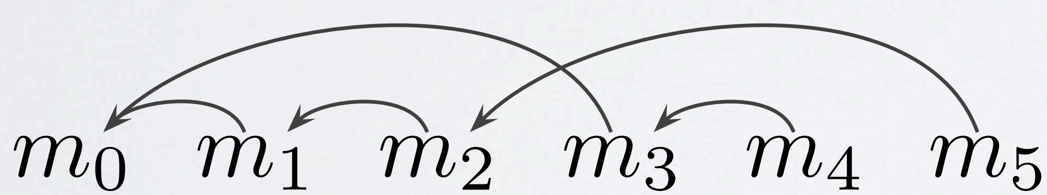
COMPOSITIONAL INTERPRETATION

- The game-semantic denotations are obtained compositionally by induction on term structure.
- Free identifiers are interpreted by identity strategies.
- All other cases are handled through composition with suitably-crafted strategies.

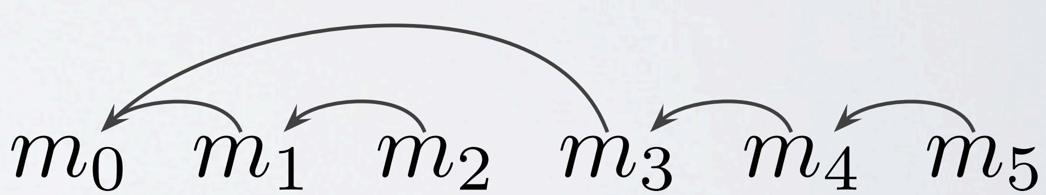
POINTERS (CBN)

$f : (\mathsf{int} \to \mathsf{int}) \to \mathsf{int}$

$f(\lambda x^{\mathsf{int}}.f(\lambda y^{\mathsf{int}}.x))$



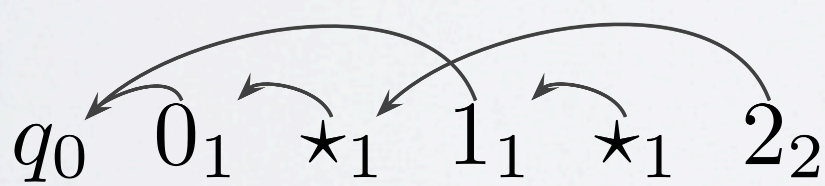
$f(\lambda x^{\mathsf{int}}.f(\lambda y^{\mathsf{int}}.y))$



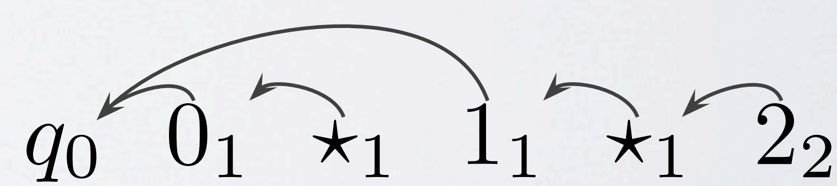
POINTERS (CBV)

f : int $\rightarrow$ int $\rightarrow$ int

let val $g = f(0)$ in
  let val $h = f(1)$ in $g(2)$



let val $g = f(0)$ in
  let val $h = f(1)$ in $h(2)$



FULL ABSTRACTION

M and N are ***contextually equivalent*** if and only if they induce the same sets of *complete* plays (all questions must be answered).

Samson Abramsky, Guy McCusker: Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. Electr. Notes Theor. Comput. Sci. 3: 2-14 (1996)

EXAMPLE (O'HEARN)

Idealized Algol: an applied lambda calculus over **com**, **int** and **var** with call-by-name evaluation and fixed-point combinators.

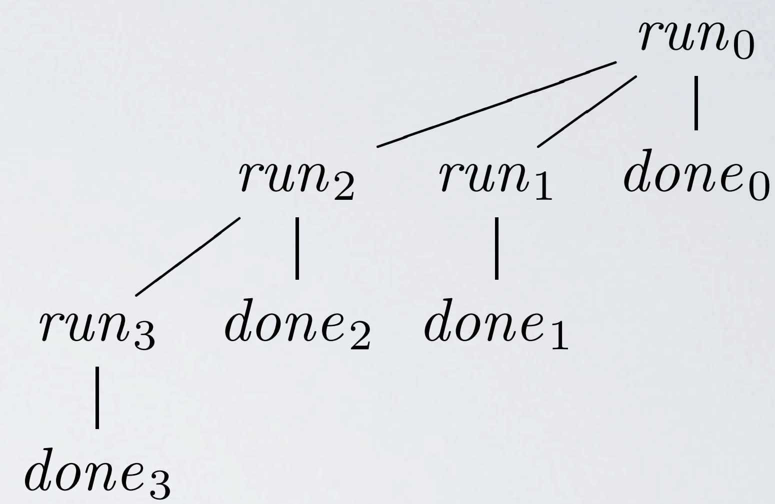
$$p : \mathbf{com} \to \mathbf{com} \vdash \; p(\Omega) : \mathbf{com}$$

$$\begin{aligned} p : \mathbf{com} \to \mathbf{com} \vdash \; \mathbf{new}\, x \,\mathbf{in} \; & x := 0; \\ & p(x := 1); \\ & \mathbf{if}\; x = 0 \;\mathbf{then\ skip\ else}\; \Omega : \mathbf{com} \end{aligned}$$

The equivalence of the two terms cannot be validated using state-transformer semantics.

GAME-SEMANTIC ARGUMENT

com is interpreted by



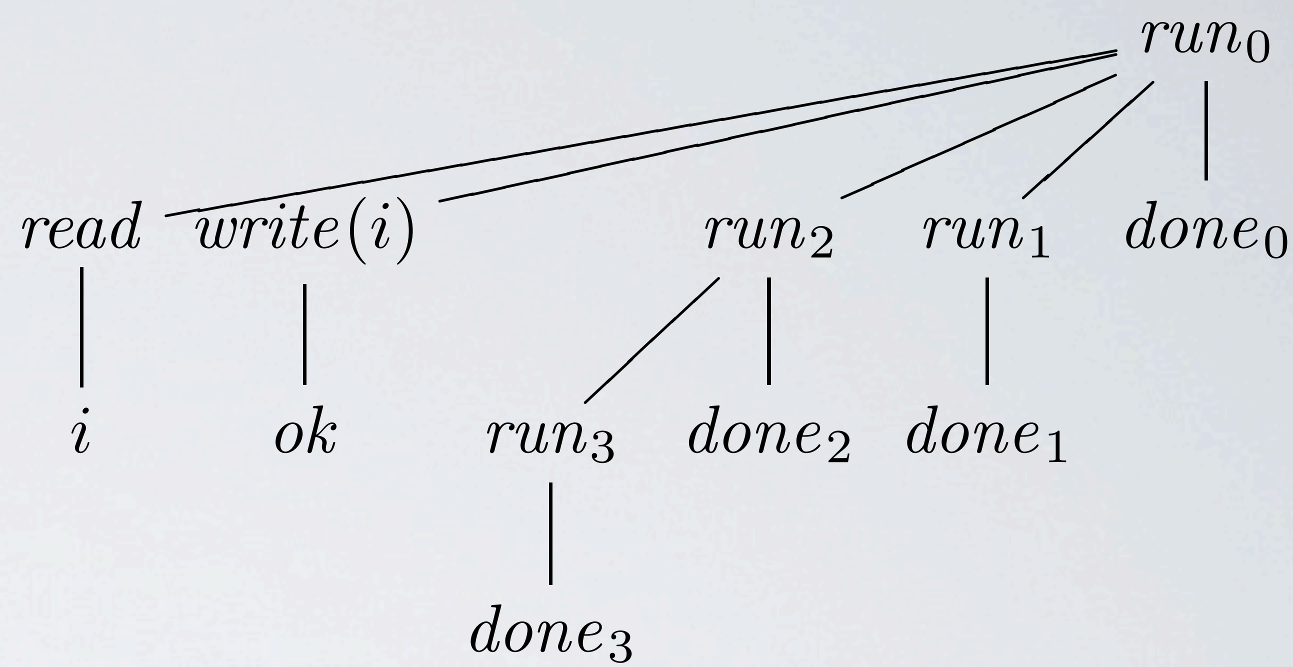
$$p : \mathbf{com}_4 \to \mathbf{com}_2 \vdash p : \mathbf{com}_1 \to \mathbf{com}_0$$

$$run_0\ run_2\ (run_3\ run_1\ done_1\ done_3)^*\ done_2\ done_0$$

$$p : \mathbf{com}_4 \to \mathbf{com}_2 \vdash p(\Omega) : \mathbf{com}_0$$

$$run_0\ run_2\ done_2\ done_0$$

$$p(x := 1)$$



- p

$$run_0\ run_2\ (run_3\ run_1\ done_1\ done_3)^*\ done_2\ done_0$$

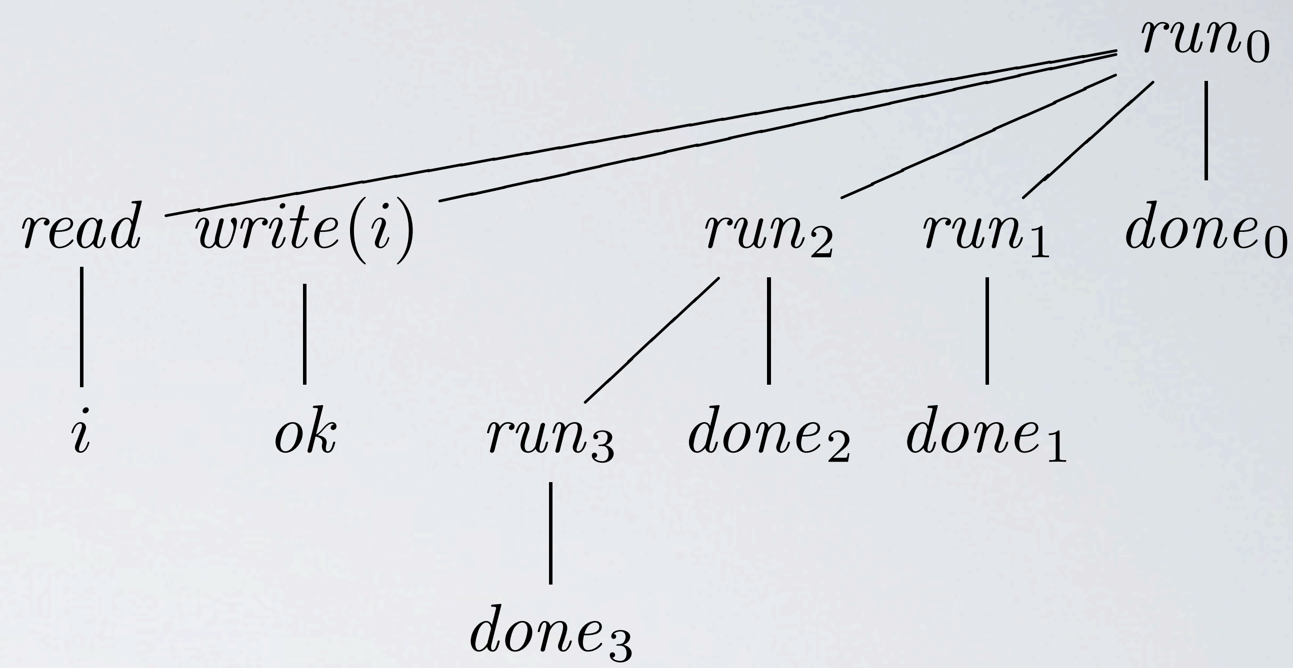
- $x := 1$

$$run_0\ write(1)\ ok\ done_0$$

- $p(x := 1)$

$$run_0\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ done_0$$

$p(x := 1)$



- p

$$run_0\ run_2\ (run_3\ run_1\ done_1\ done_3)^*\ done_2\ done_0$$

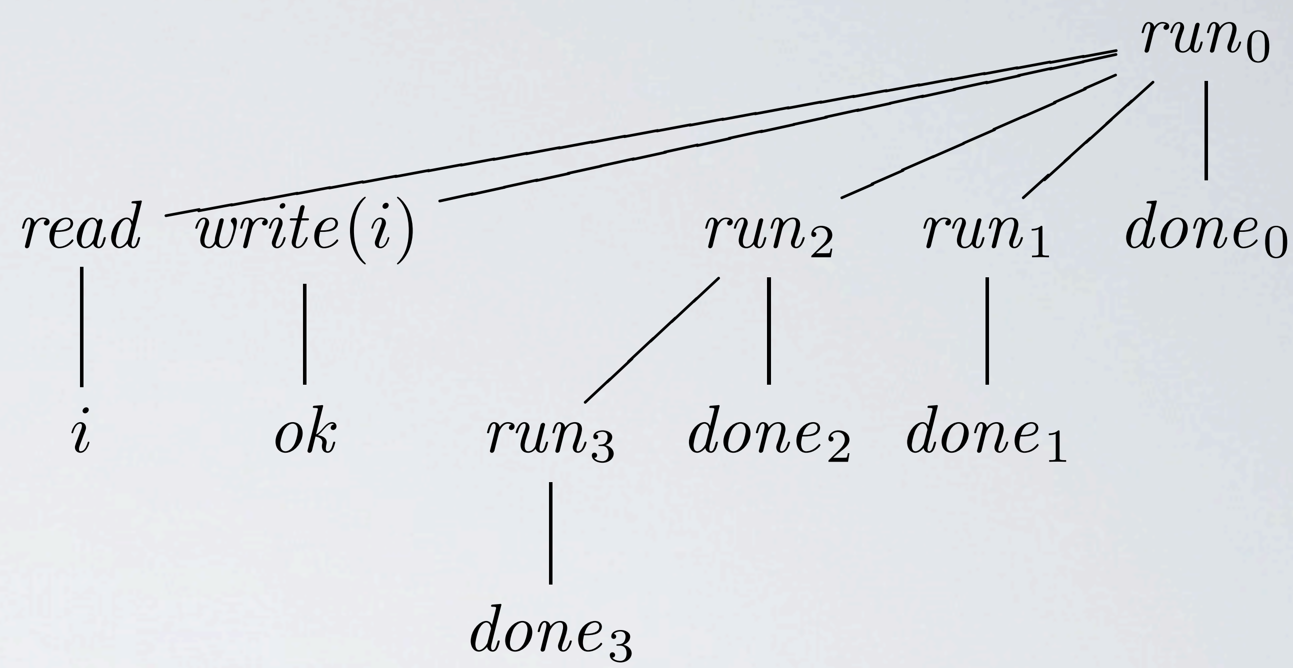
- $x := 1$

$$run_0\ write(1)\ ok\ done_0$$

- $p(x := 1)$

$$run_0\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ done_0$$

$$p(x := 1)$$



- p

$$run_0\ run_2\ (run_3\ run_1\ done_1\ done_3)^*\ done_2\ done_0$$

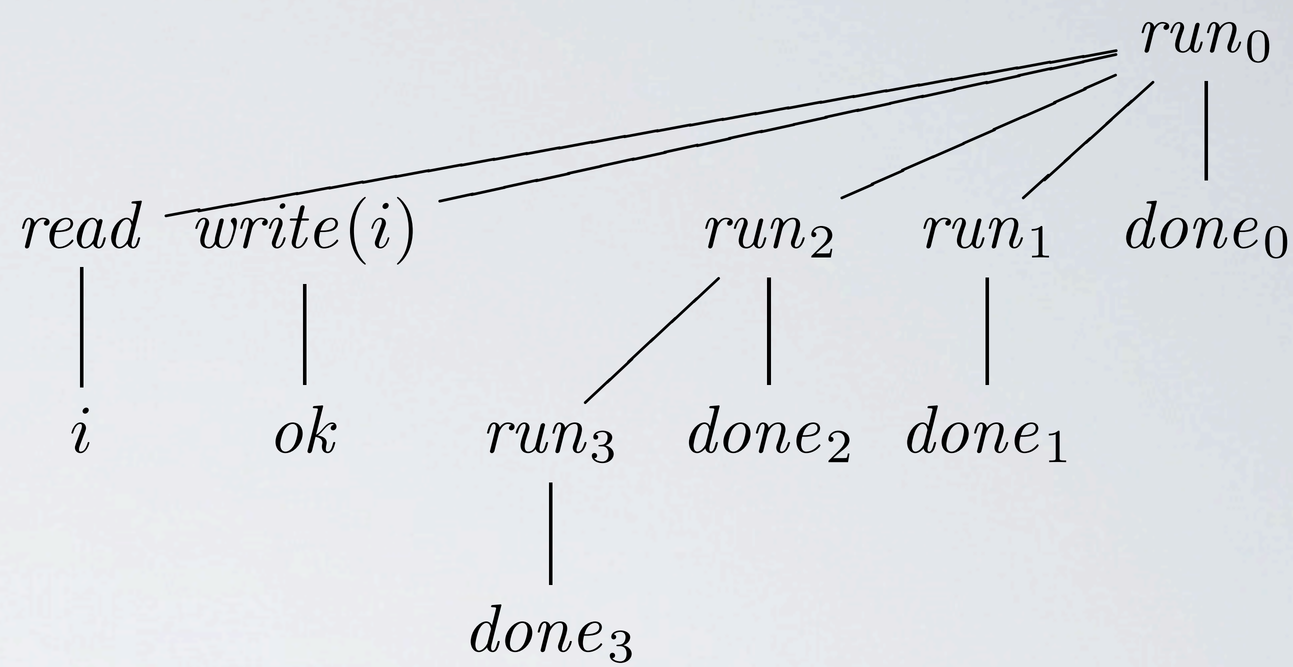
- $x := 1$

$$run_0\ write(1)\ ok\ done_0$$

- $p(x := 1)$

$$run_0\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ done_0$$

$p(x := 1)$



- p

$$run_0\ run_2\ (run_3\ run_1\ done_1\ done_3)^*\ done_2\ done_0$$

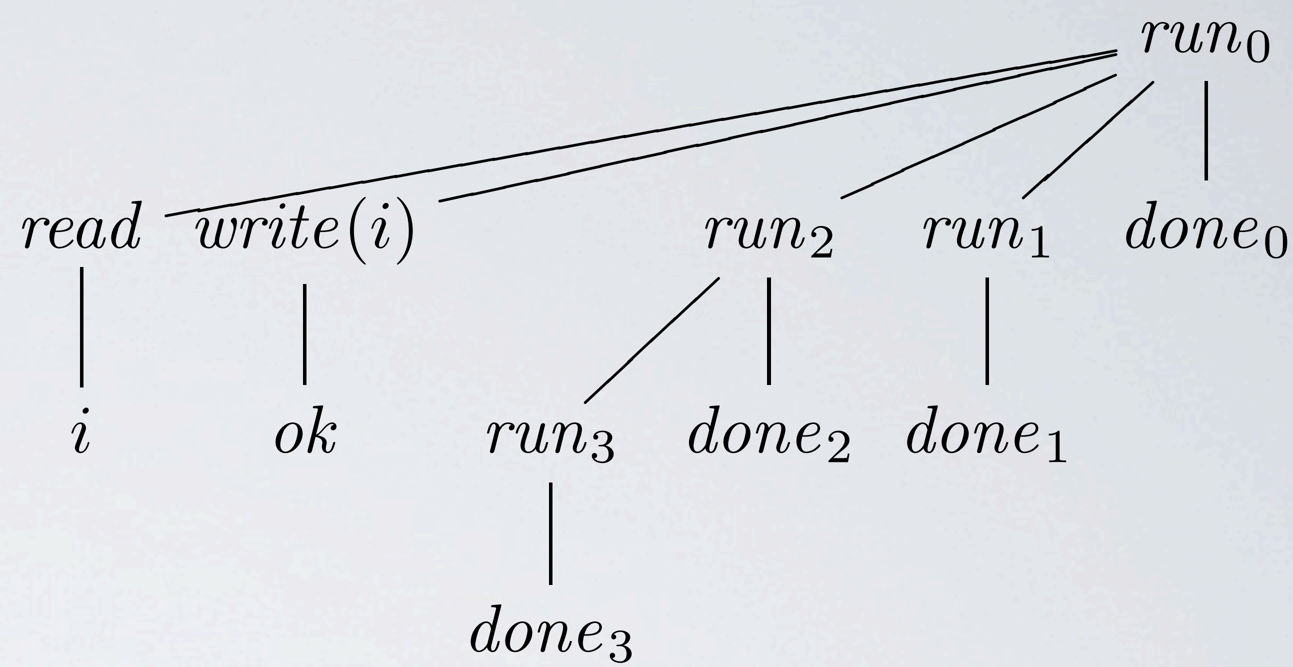
- $x := 1$

$$run_0\ write(1)\ ok\ done_0$$

- $p(x := 1)$

$$run_0\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ done_0$$

$$p(x := 1)$$



- p

$$run_0 \; run_2 \; (run_3 \; run_1 \; done_1 \; done_3)^* \; done_2 \; done_0$$

- $x := 1$

$$run_0 \; write(1) \; ok \; done_0$$

- $p(x := 1)$

$$run_0 \; run_2 \; (run_3 \; write(1) \; ok \; done_3)^* \; done_2 \; done_0$$

- $p(x := 1)$

$$run_0\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ done_0$$

- $x := 0;\ p(x := 1);\ \mathbf{if}\ x = 0\ \mathbf{then}\ ()\ \mathbf{else}\ \Omega$

$$run_0\ write(0)\ ok\ run_2\ (run_3\ write(1)\ ok\ done_3)^*\ done_2\ read\ 0\ done_0$$

- $\mathbf{new}\ x\quad \mathbf{in}\quad x := 0; p(x := 1); \mathbf{if}\ x = 0\ \mathbf{then}\ ()\ \mathbf{else}\ \Omega$

$$run_0\ run_2\ done_2\ done_0$$

new is interpreted by composition with a strategy ensuring that *read*'s and $write(i)$'s match.

Same complete plays imply equivalence.

RECIPE

- Analyze the underlying process of composition.
- Understand what “really happens”.
- Express strategy-building in a concrete way as an operation on formal languages.
- Remember to encode pointers, if necessary.
- Prove language equivalence using the chosen representation.

TYPE ORDER

$$\mathsf{ord}(\theta) = \begin{cases} 0 & \theta \equiv \mathbf{com}, \mathbf{int}, \mathbf{var} \\ \max(\mathsf{ord}(\theta_1) + 1, \mathsf{ord}(\theta_2)) & \theta \equiv \theta_1 \to \theta_2 \end{cases}$$

- IA_k consists of terms of the form

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

  with $\mathsf{ord}(\theta_i) < k$ and $\mathsf{ord}(\theta) \leq k$.

- Looping and recursion are not available in IA_k.

- We write $\mathbf{Y}_k$ to stress the availability of the fixed-point combinator $\mathbf{Y}_\theta : (\theta \to \theta) \to \theta$ for θ of order k.

DECIDABILITY

We assume finite ground types!

	pure	**+while**	$+\mathbf{Y}_0$	$+\mathbf{Y}_1$
IA_1	+	+	+	−
IA_2	+	+	+	−
IA_3	+	+	+	−
IA_4	−	−	−	−

The results were obtained using FA, DPDA and VPA.

BIBLIOGRAPHY

FA

Dan R. Ghica, Guy McCusker: Reasoning about Idealized ALGOL Using Regular Languages. ICALP 2000: 103-115

Andrzej S. Murawski: Games for complexity of second-order call-by-name programs. Theor. Comput. Sci. 343(1-2): 207-236 (2005)

DPDA

C.-H. Luke Ong: Observational Equivalence of 3rd-Order Idealized Algol is Decidable. LICS 2002: 245-256

Andrzej S. Murawski, C.-H. Luke Ong, Igor Walukiewicz: Idealized Algol with Ground Recursion, and DPDA Equivalence. ICALP 2005: 917-929

VPA

Andrzej S. Murawski, Igor Walukiewicz: Third-Order Idealized Algol with Iteration Is Decidable. FoSSaCS 2005: 202-218

undecidability

Andrzej S. Murawski: On Program Equivalence in Languages with Ground-Type References. LICS 2003: 108-

COMPLEXITY

Equivalence of terms in beta-normal form.

	pure	+**while**	$+\mathbf{Y}_0$	$+\mathbf{Y}_1$
IA_1	coNP-complete	PSPACE-complete	?	—
IA_2	PSPACE-complete	PSPACE-complete	?	—
IA_3	EXPTIME-complete	EXPTIME-complete	?	—
IA_4	—	—	—	—

Non-elementary in general.

UNDECIDABILITY

- It may seem surprising that program equivalence in a language over finite datatypes is undecidable.
- This is all due to the rich structure of interactions afforded by higher-order types.
- At fourth order there are patterns of interaction between O and P that resemble actions of a queue.
- Moreover, there exists a program that can detect whether O follows the queue-pattern.
- Game semantics tames higher-order interaction.

NONDETERMINISM

May-termination $\Downarrow_{\text{may}}$

Must-termination $\Downarrow_{\text{must}}$

- May-equivalence

$$\forall \mathbb{C}[-]. \qquad \mathbb{C}[M] \Downarrow_{\text{may}} \quad \Longleftrightarrow \quad \mathbb{C}[N] \Downarrow_{\text{may}}$$

- Must-equivalence

$$\forall \mathbb{C}[-]. \qquad \mathbb{C}[M] \Downarrow_{\text{must}} \quad \Longleftrightarrow \quad \mathbb{C}[N] \Downarrow_{\text{must}}$$

- May & Must-equivalence

MAY-EQUIVALENCE

Characterization via complete plays still applies.

	pure	**+while**	$+\mathbf{Y}_0$
EA_1	PSPACE-complete	EXPSPACE-complete	–
EA_2	EXPSPACE-complete	EXPSPACE-complete	–
EA_3	2-EXPTIME-complete	2-EXPTIME-complete	–
EA_4	–	–	–

MUST-EQUIVALENCE

Russell Harmer, Guy McCusker: A Fully Abstract Game Semantics for Finite Nondeterminism. LICS 1999: 422-430

A ***strategy*** σ on an arena A is a pair (T_σ, D_σ). The first component T_σ, known as the ***traces*** of σ, is a non-empty set of even-length legal plays of A satisfying

$$sab \in T_\sigma \Rightarrow s \in T_\sigma .$$

We write $\mathsf{dom}(\sigma)$ for the ***domain*** of σ, *i.e.* the set $\{sa \in L_A \mid \exists b.\, sab \in T_\sigma\}$ and $\mathsf{cc}(\sigma)$ for the ***contingency closure*** of σ, *i.e.* $T_\sigma \cup \mathsf{dom}(\sigma)$. Given $sa \in \mathsf{dom}(\sigma)$, let $\mathsf{rng}_\sigma(sa) = \{b \in M_A \mid sab \in T_\sigma\}$.

The second component D_σ is known as the ***divergences*** of σ; it's a set of odd-length legal plays of A satisfying

Characterization via quotienting.

WINNING REGIONS

Let O and P play a *reachability* game over the traces of σ. O will be declared a winner if he reaches a complete play without encountering any divergences. This induces winning regions for O and P.

Two terms are *must-equivalent* if and only if any difference between the induced strategies (trace or divergence) is compensated by a winning region for P.

Andrzej S. Murawski: Reachability Games and Game Semantics: Comparing Nondeterministic Programs. LICS 2008: 353-363

MUST-EQUIVALENCE

	pure	+**while**	$+\mathbf{Y}_0$
EA_1	PSPACE-complete	2-EXPTIME-complete	–
EA_2	2-EXPTIME-complete	2-EXPTIME-complete	–
EA_3	3-EXPTIME-complete	3-EXPTIME-complete	–
EA_4	–	–	–

PROBABILISTIC EQUIVALENCE

$$\Downarrow_p$$

$$\forall\, \mathbb{C}[-]. \qquad \mathbb{C}[M] \Downarrow_p \quad \Longleftrightarrow \quad \mathbb{C}[N] \Downarrow_p$$

PROBABILISTIC STRATEGIES

The definition comes in two steps. First of all, we define a ***prestrategy*** σ on an arena A to be a (set-theoretic) function $\sigma : \mathcal{L}_A^{\mathsf{even}} \to [0, \infty]$. Such a prestrategy is a ***strategy*** iff

(**p1**) $\sigma(\varepsilon) = 1$;

(**p2**) if $sa \in \mathcal{L}_A^{\mathsf{odd}}$ then $\sigma(s) \geq \sum_{t \in \mathsf{ie}(sa)} \sigma(t)$.

Vincent Danos, Russell Harmer: Probabilistic game semantics.
ACM Trans. Comput. Log. 3(3): 359-382 (2002)

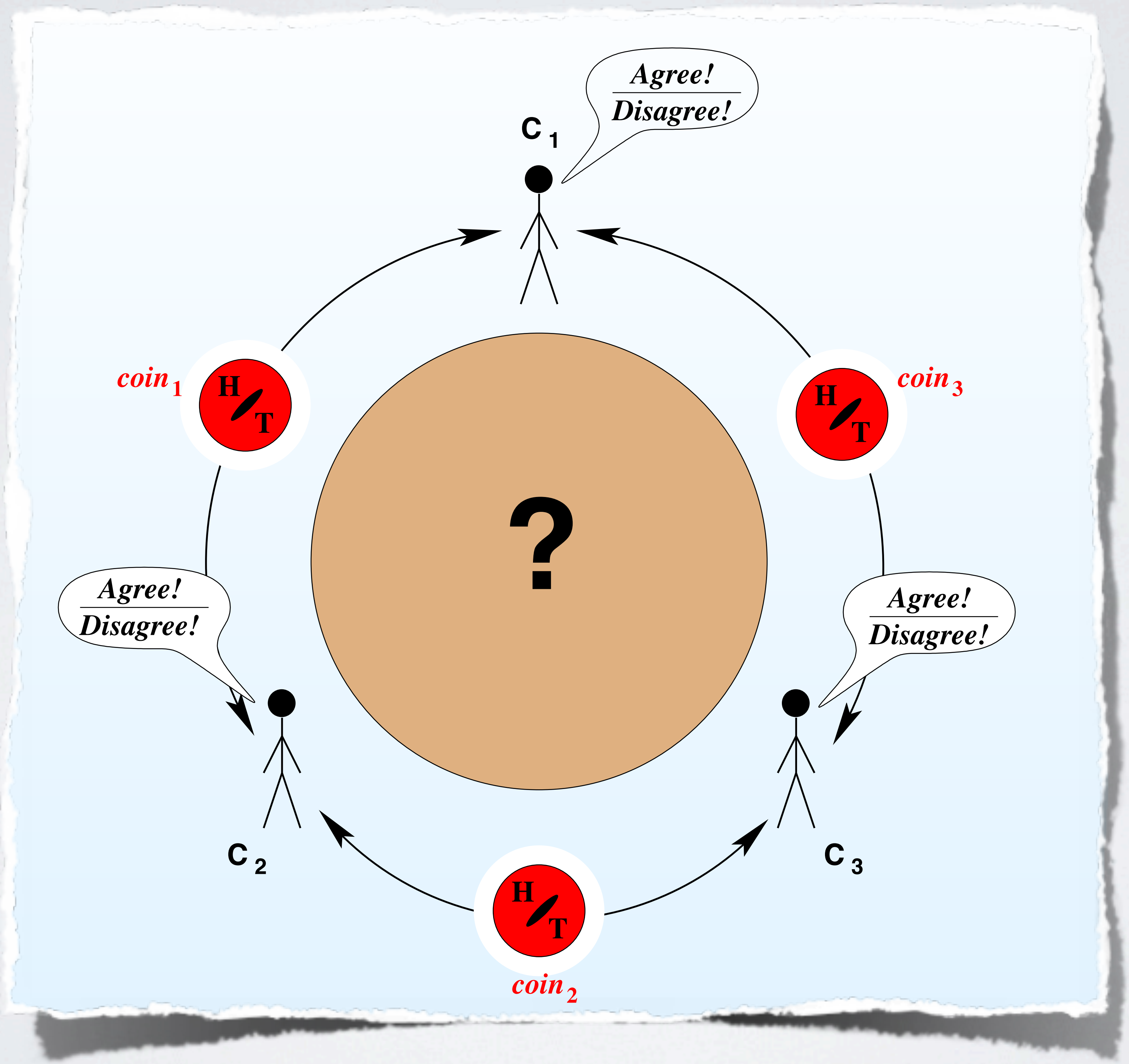
PROBABILISTIC LANGUAGE EQUIVALENCE

Two probabilistic programs are *equivalent* if and only if the corresponding probabilistic strategies assign the same probabilities to all complete plays.

APEX tool

Axel Legay, Andrzej S. Murawski, Joël Ouaknine, James Worrell: On Automated Verification of Probabilistic Programs. TACAS 2008: 173-187

DINING CRYPTOGRAPHERS



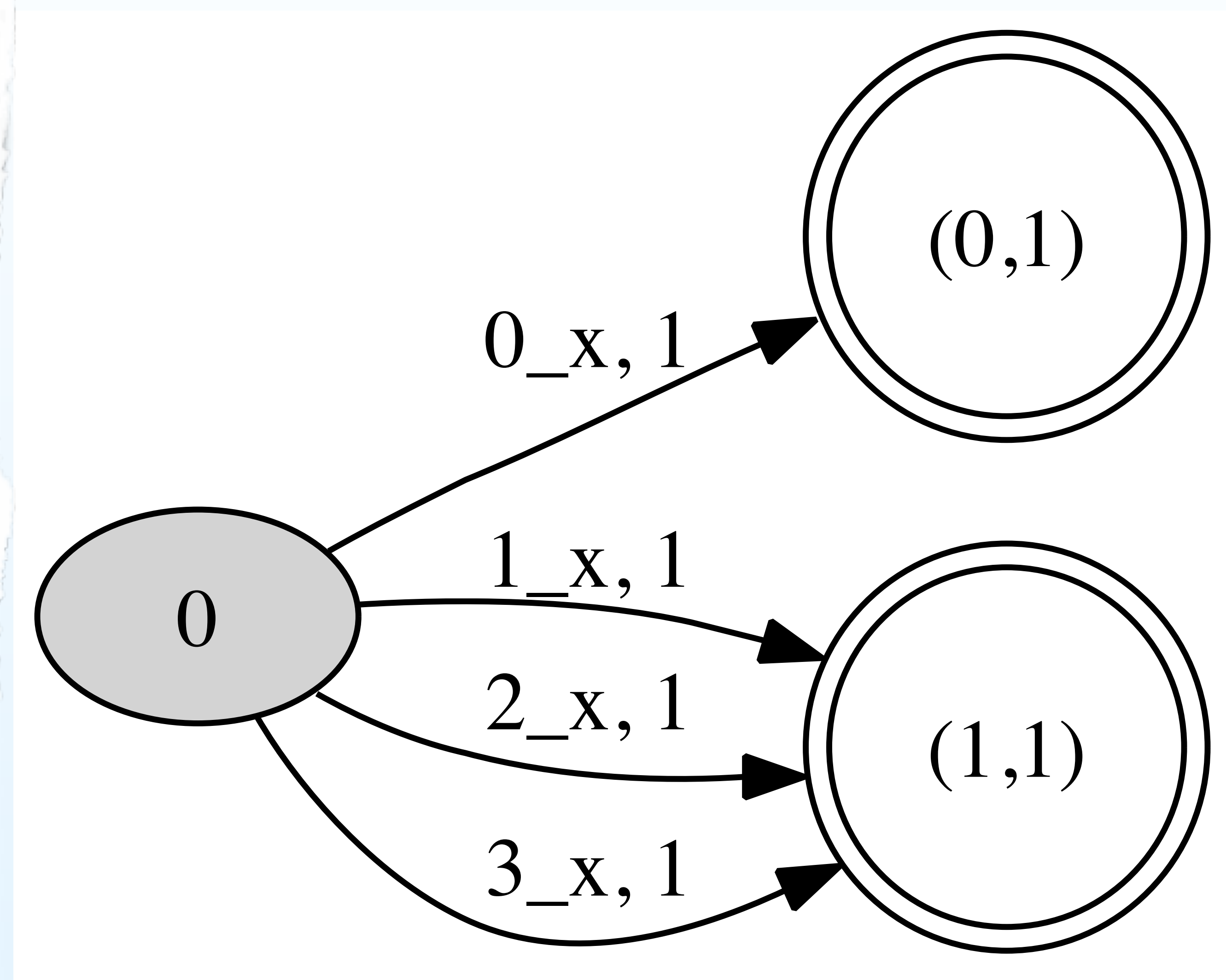
WAS IT ONE OF THEM?

One of the cryptographers paid $\iff$ #“Disagree” is odd

$$f : \{0, 1, 2, 3\} \to \{0, 1\}$$

```
x:int%4 |-
  var%4 whopaid; var%2 first; var%2 left;
  var%2 right; var%2 parity; var%4 i;

  whopaid:=x; first:=coin; right:=first; i:=1;

  while (i) do
  {
    left:= if (i=3) then first else coin;
    if not((left=right)+(whopaid=i))
                then parity:=not(parity);
    right := left;
    i:=i+1
  };
  parity : int%2
```

CORRECTNESS

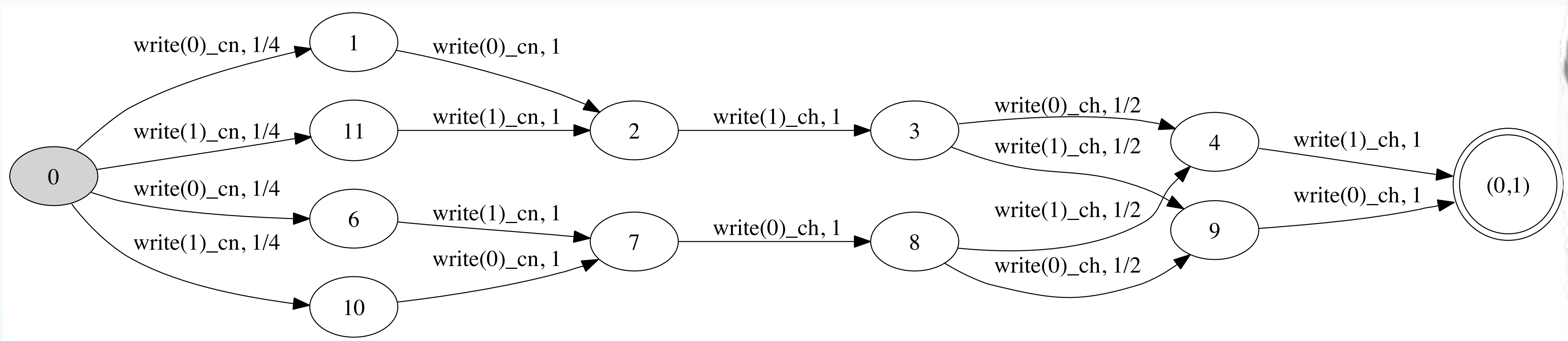


ANONYMITY (VIEWS)

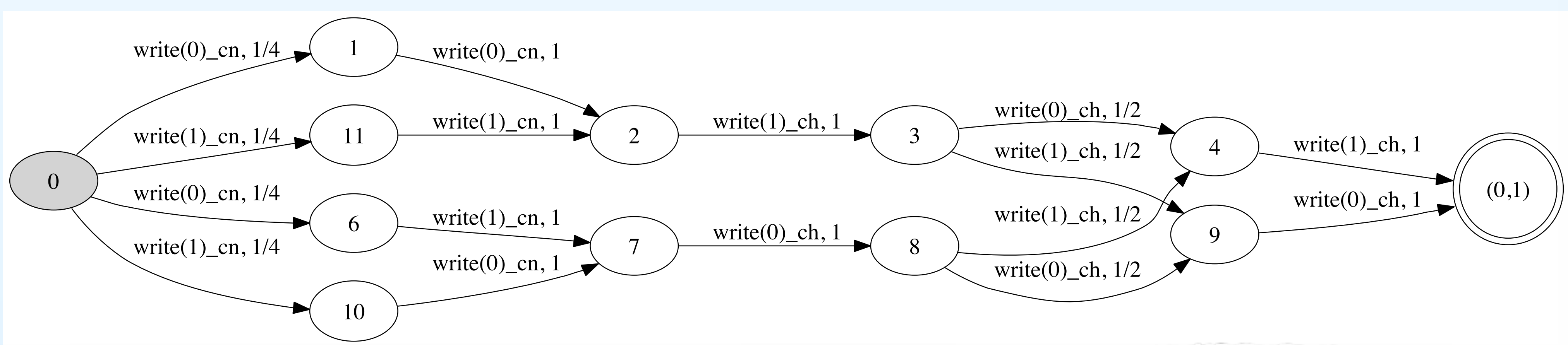
```
cn:var%2, ch:var%2 |-

var%4 whopaid;
whopaid := 2;

if (whopaid <= 1) then diverge else
{
  var%2 first; var%2 left; var%2 right; var%4 i;

  first:=coin; right:=first; i:=1;

  while (i) do
  {
    left:=if (i=3) then first else coin;
    if (i=1) then { cn:=right; cn:=left };
    if ((left=right)+(whopaid=i)) then ch:=1 else ch:=0;
    right := left;
    i := i+1
  }
}: com
```

WHAT CAN HE SEE?

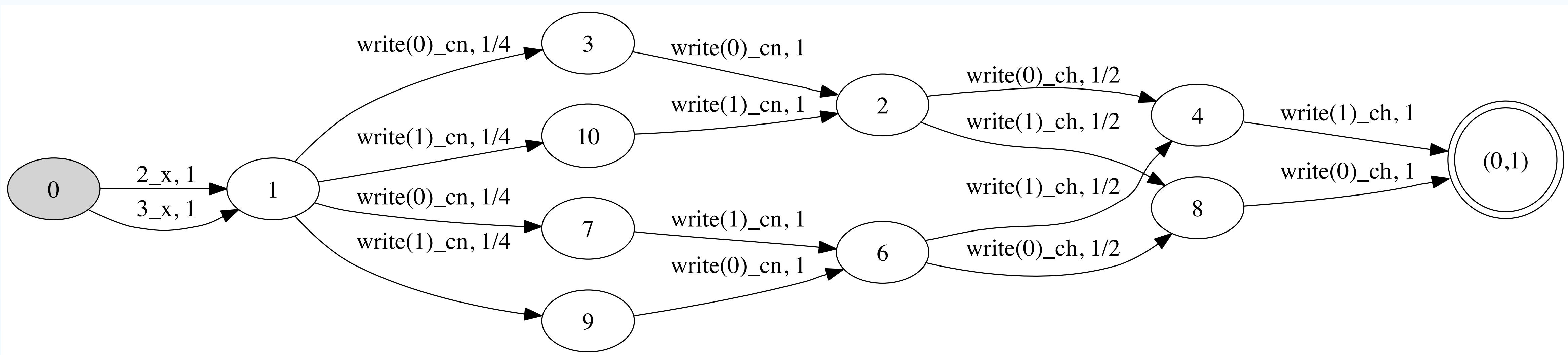
2 paid



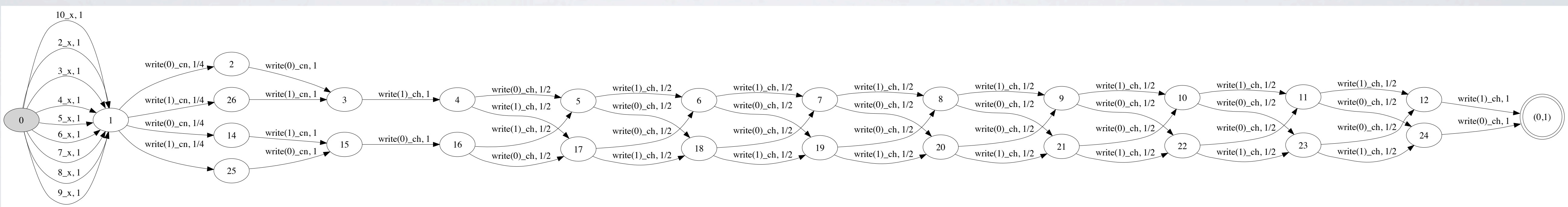
3 paid



WHAT CAN HE SEE?



MORE CRYPTOGRAPHERS



OTHER TOOLS

- Homer

David Hopkins, C.-H. Luke Ong: Homer: A Higher-Order Observational Equivalence Model checkER. CAV 2009: 654-660

- MAGE

Adam Bakewell, Dan R. Ghica: On-the-Fly Techniques for Game-Based Software Model Checking. TACAS 2008: 78-92

CALL-BY-VALUE EVALUATION

Call-by-value Idealized Algol

RML: an ML-like language with integer references, including "bad" ones

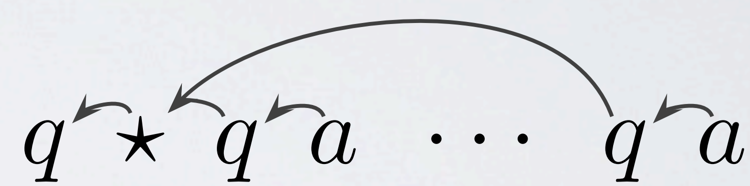
$$\mathsf{ref\ int} = (\mathsf{unit} \to \mathsf{int}) \times (\mathsf{int} \to \mathsf{unit})$$

PRO: Finite alphabet, if finitely many values!
CON: Equivalences relying on ref int may be affected.

Samson Abramsky, Guy McCusker: Call-by-Value Games. CSL 1997: 1-17

SOME SURPRISES?

– $\mathsf{unit} \to \mathsf{unit} \to \mathsf{unit}$ is problematic.



There are many a's to point at...

– $(\mathsf{unit} \to \mathsf{unit}) \to (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{unit}$ is undecidable.

Andrzej S. Murawski: Functions with local state: Regularity and undecidability. Theor. Comput. Sci. 338(1-3): 315-349 (2005)

SOME RESULTS

Assume finite ground types and absence of recursion.

- Regular

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

- Visibly context-free

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

David Hopkins, Andrzej S. Murawski, C.-H. Luke Ong: A Fragment of ML Decidable by Visibly Pushdown Automata. ICALP (2) 2011: 149-161

SUMMARY

- Many decision procedures have been obtained via game semantics in recent years.
- Some have been implemented and observed to beat alternative approaches.
- Several tools use game semantics as a main engine.
- Ready for "realistic" applications?