Computabilità, Complessità e Logica

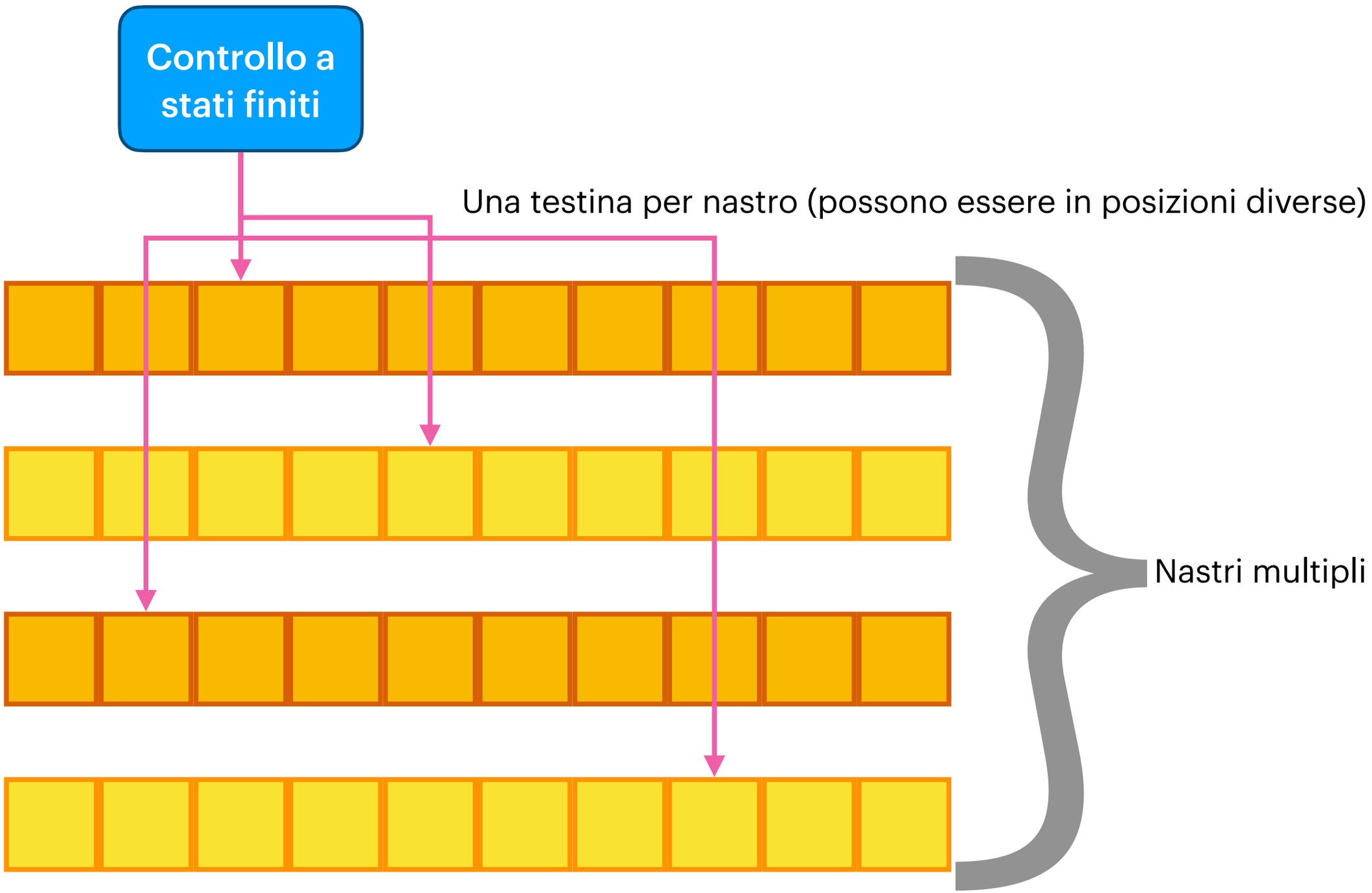
Lezione 9

Modelli universali

Le macchine di Turing non sono l'unico modello di calcolo

- Macchine di Turing multi-nastro
- Macchine di Turing non deterministiche
- Macchine a registri
- Lambda calcolo
- La maggior parte dei linguaggi di programmazione
- Sistemi a membrane, Combinatori SKI,
  Magic: The Gathering™, etc.

MdT multi-nastro



MdT multi-nastro

Una MdT con k nastri è una settupla
$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

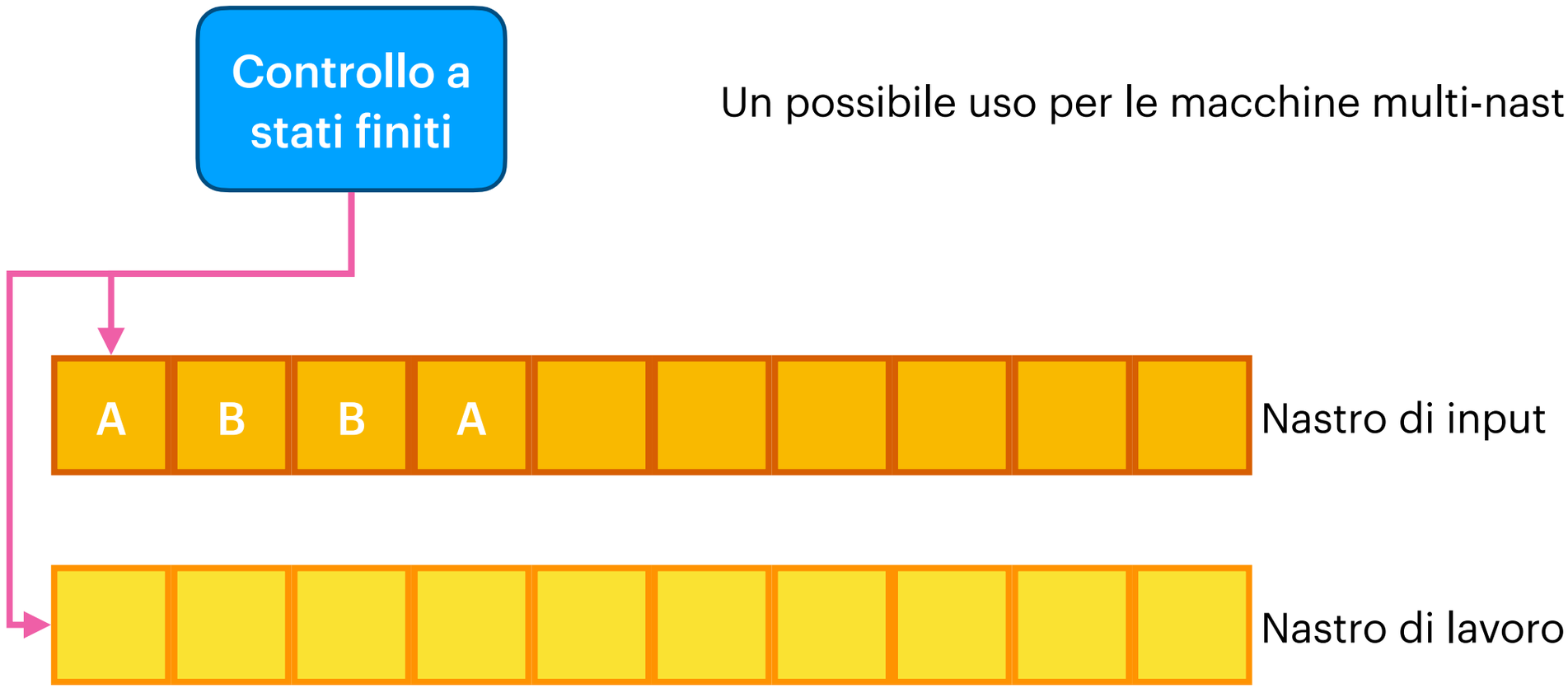
- Cambia solo la definizione della funzione di transizione, che deve indicare cosa deve fare ognuna delle k testine

- La funzione di transizione è definita come:
  $\delta : Q \times \Gamma^k \to Q \times (\Gamma \times \{ \leftarrow , \rightarrow \})^k$

  - Vengono letti e scritti k simboli (uno per testina)

  - Ognuna delle testine si muove in modo indipendente

MdT multi-nastro



Un possibile uso per le macchine multi-nastro

In questo modo possiamo contare quanto spazio usa la macchina in aggiunta all'input

Oppure possiamo semplificare la scrittura di algoritmi direttamente usando la MdT

MdT non deterministiche

E se la macchina di Turing non avesse una sola “mossa” possibile?

Come per gli automi non deterministici possiamo cambiare la funzione di transizione:

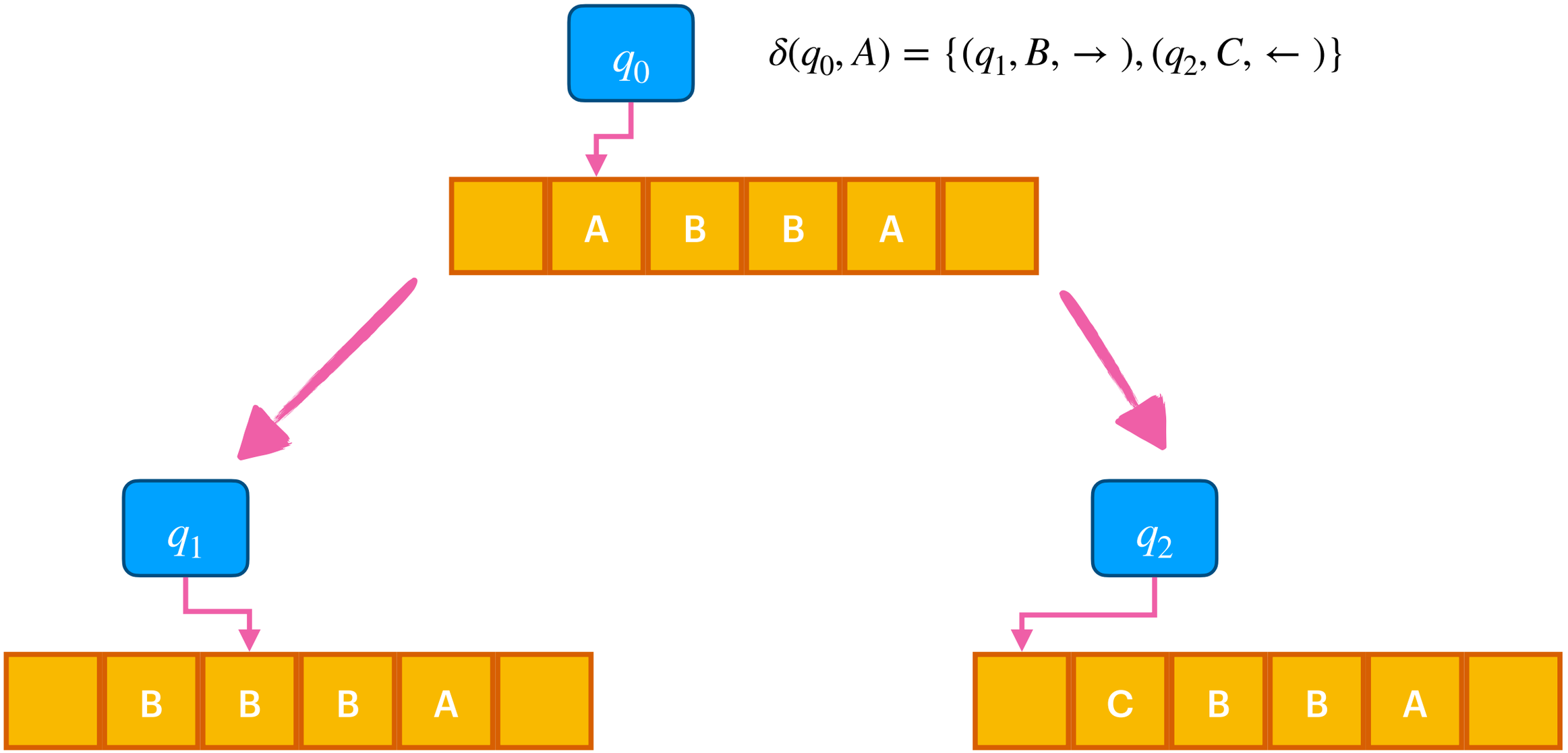
Se nello stato $q \in Q$ leggiamo il simbolo $\sigma \in \Gamma$...

..invece di una tripla $(q', \sigma', d) \in Q \times \Gamma \times \{ \leftarrow , \rightarrow \}$...

...abbiamo un sotto-insieme di $Q \times \Gamma \times \{ \leftarrow , \rightarrow \}$...

...ognuna rappresentante una possibile scelta non deterministica della MdT

MdT non deterministiche



Abbiamo due possibili scelte e quindi due possibili computazioni

Non vi è più una unica computazione per ogni input!

MdT non deterministiche

Una MdT non deterministica è una settupla
$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

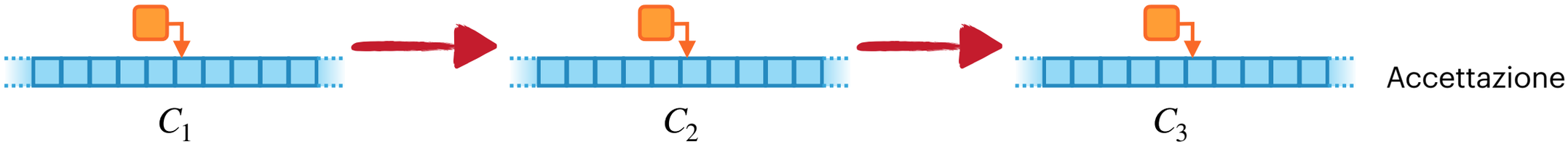
Rispetto alla definizione di macchina di Turing standard cambia solo la definizione della funzione di transizione

Per le macchine non deterministiche
$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\leftarrow, \rightarrow\}}$

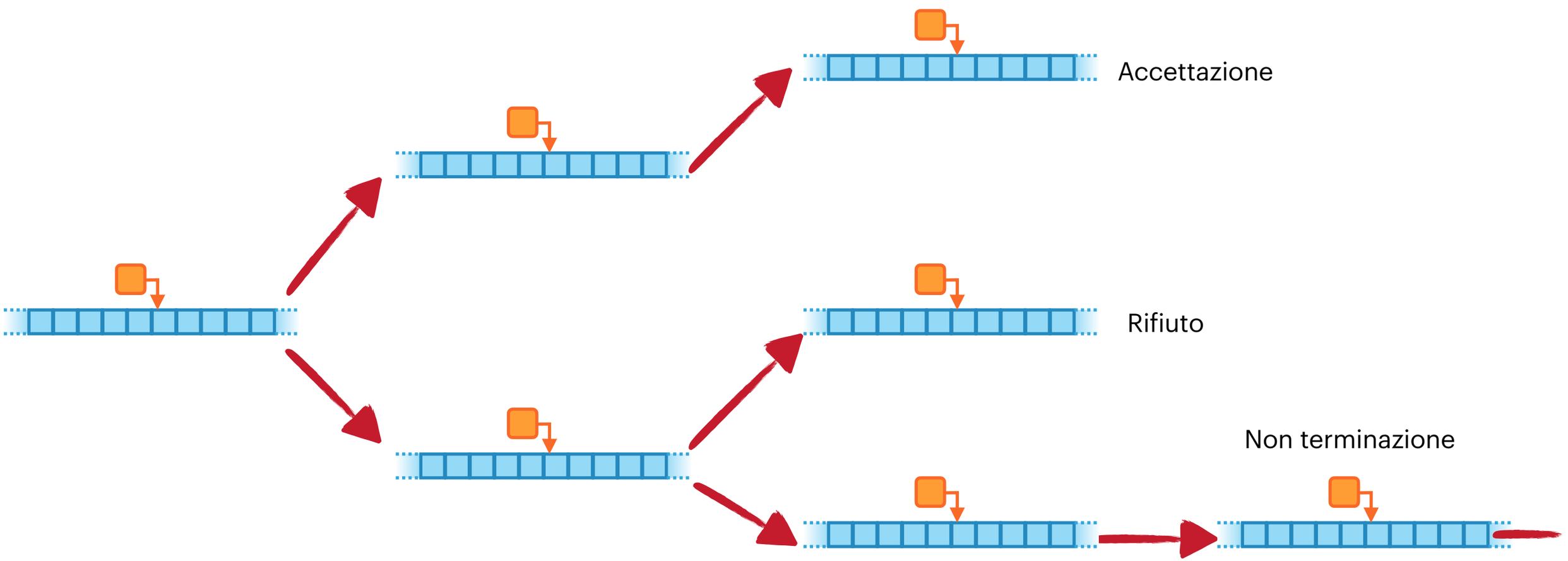
Ne segue che $\delta(q, \sigma)$ per $q \in Q$ e $\sigma \in \Gamma$ è un **insieme** di triple e quindi di possibili transizioni che la macchina può compiere

MdT non deterministiche

Macchina deterministica



Macchina non deterministica



MdT non deterministiche

Definire la nozione di una accettazione di una MdT non deterministica non è banale

Nel caso deterministico serve che la computazione si arresti e si sia nello stato accettante

Nel caso di MdT non deterministiche può esserci più di una computazione...

...e non è detto che tutte le computazioni rispondano allo stesso modo.

Per convenzione diciamo che una macchina non deterministica in cui tutte le computazioni si arrestano accetta se **esiste** una computazione accettante

MdT non deterministiche

Una MdT non deterministica non corrisponde a nessun dispositivo di calcolo fisicamente realizzabile

Possiamo pensare a MdT non deterministiche come macchine a “parallelismo infinito”:

- Se si può scegliere tra due opzioni il programma fa una fork() e si duplica
- La prima copia sceglie di fare la prima mossa e prosegue
- La seconda copia sceglie di fare la seconda mossa e prosegue

Possiamo comunque simulare una MdT non-deterministica con una deterministica

Macchine a registri

È possibile pensare a una macchina a registri come un normale processore con n registri di capienza infinita e un insieme di istruzioni molto limitato.

Conta solo il valore contenuto nei registri

Se abbiamo un numero sufficiente di registri non ci serve memoria (possiamo salvare tutto nei registri)

Vediamo una variante con solo due istruzioni possibili

Questa variante con un numero sufficiente di registri è universale

Macchine a registri

Una macchina a registri contiene n registri chiamati $r_1, r_2, \ldots, r_n$. Ciascuno di questi può contenere un qualsiasi elemento di $\mathbb{N}$

Un programma è una sequenza di m istruzioni. Ogni istruzione ha una associata etichetta $l_1, l_2, \ldots, l_m$

Le due istruzioni possibili sono della forma:

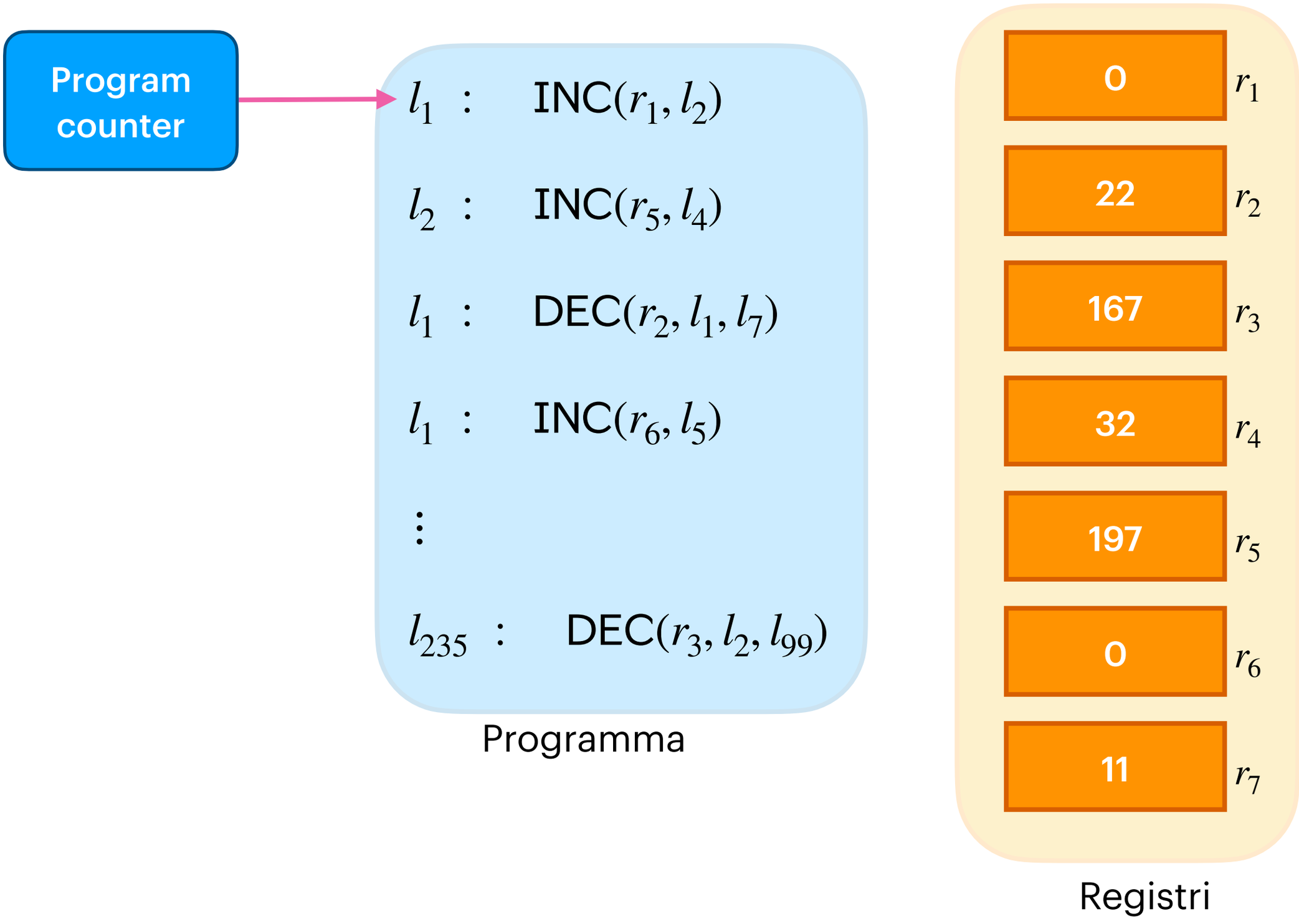
- Incremento + salto non condizionato
- Decremento se non zero + salto a seconda del valore nel registro

Macchine a registri

Istruzioni:

- INC(r_i, l_j) incrementa di 1 il valore nel registro r_i e salta all'istruzione con etichetta l_j
- DEC(r_i, l_j, l_k) se non zero decrementa di 1 il valore del registro r_i e salta all'istruzione con etichetta l_j. Se il valore nel registro r_i è zero salta invece all'istruzione con etichetta l_k

Macchine a registri



Gödelizzazione

Una breve digressione: con un numero limitato di registri come possiamo tenere un numero arbitrariamente elevato di numeri piccoli?

Un metodo comune è la Gödelizzazione, un modo di codificare più numeri in un solo numero ideato da Kurt Gödel negli anni ’30

Idea: sfruttare il fatto che numero ha una unica fattorizzazione in fattori primi

Gödelizzazione

Supponiamo di voler codificare la sequenza $x_1, \ldots, x_m$

Siano $p_1, \ldots, p_m$ i primi m numeri primi

Il numero $p_1^{x_1} \times p_2^{x_2} \times \cdots \times p_m^{x_m}$ identifica unicamente la sequenza $x_1, \ldots, x_m$ (incluso l'ordine)

Esempio: scriviamo il numero di Gödel di $4, 7, 6$

Primi tre numeri primi: $2,3,5$

Codifica: $2^4 \times 3^7 \times 5^6 = 273375000$

Non una codifica pratica ma utile in diverse dimostrazioni

Lambda calcolo

Ideato da Alonzo Church negli anni ’30

Nella sua forma di base consiste di tre modi di costruire dei **lambda termini**:

- x il nome di una variabile è un lambda-termine
- $(\lambda x\,.\,M)$ dove M è un lambda termine e x un nome di variabile. Questa è l’*astrazione*, che definisce una funzione ove la variabile x diventa legata nel corpo di M
- $(M\ N)$ dove M e N sono lambda termini. Questa è l’*applicazione*, dove una funzione è applicata a un argomento

Lambda calcolo

Esempi di lambda-termini:

- x
- $\lambda x \,.\, x$
- $\lambda x \,.\, (\lambda y \,.\, (\lambda z \,.\, ((z\ y)\ x)))$

La computazione avviene tramite l'applicazione ripetuta di regole che riscrivono la formula fino a quando non è possibile applicare alcuna altra operazione

Lambda calcolo

Operazioni di riduzione

- α-conversione. Rinomina una variabile per evitare collisioni di nomi. E.g., da $\lambda x \,.\, x$ a $\lambda y \,.\, y$
- β-riduzione. Cattura la nozione di applicazione di una funzione. È definita la riscrittura di $(\lambda x \,.\, M)\ N$ in $M[x := N]$, ovvero M dove tutte le istanze di x sono sostituite fa N.
  E.g., $(\lambda x \,.\, (\lambda y \,.\, x\ y))\ (\lambda z \,.\, z\ z)$ viene riscritto in $\lambda y \,.\, (\lambda z \,.\, z\ z)\ y$

Lambda calcolo

Operazioni di riduzione

- η-riduzione. Cattura la nozione che se una variabile non appare nel corpo di una funzione allora possiamo "estrarre" il corpo della funzione o aggiungere una funzione con una variabile "dummy". Ovvero, $\lambda x\,.\,M$ e M sono equivalenti se x non appare in M.

Questa è solo una breve panoramica di cosa sia il lambda calcolo. Non vediamo come effettivamente lavorarci, è solo per vedere cosa sia.

Magic the Gatering è Turing-completo

Link all'articolo che mostra come costruire una macchina di Turing usando Magic: The Gathering rispettando tutte le regole del gioco: https://arxiv.org/abs/1904.09828

Magic: The Gathering is Turing Complete

Alex Churchill
Independent Researcher
Cambridge, United Kingdom
alex.churchill@cantab.net

Stella Biderman
Georgia Institute of Technology
Atlanta, United States of America
stellabiderman@gatech.edu

Austin Herrick
University of Pennsylvania
Philadelphia, United States of America
aherrick@wharton.upenn.edu

***Abstract*—*Magic: The Gathering* is a popular and famously complicated trading card game about magical combat. In this paper we show that optimal play in real-world *Magic* is at least as hard as the Halting Problem, solving a problem that been open for a decade [1], [10]. To do this, we present a ... arbitrary Turing machine into a ... thodology for embedding an ... to win the game of *Magic* such that the first player is guaranteed ... halts. Our result applies ... if and only if the Turing machine ... can be achieved using standard- ... does not rely on stochasticity ... is also highly unusual in that ... ced in the construction. This ... ll win a game in which neither ... make for the rest of the game ... a discussion of the implications ... y of games and remarks about ... a tournament setting.**

... UCTION

... known as *Magic*) is a popular ... izards of the Coast. Formally, it ... astic card game with imperfect ... same category as games like ... se games, players design their ... card-pool of over 20,000 cards. ... has made it a popular topic in ...

... *Magic: The Gathering* from the ... ic game theory, looking at the ... f evaluating who will win a game. ... limits on their complexity (such ... ard) most research in algorithmic ... d games has primarily looked at the ... generalisations of commonly played games rather than the ... versions of the games. A few real-world games have ... d to have non-trivial complexity, including *Dots-and-Boxes*, *Jenga* and *Tetris* [8]. We believe that no real-world game is known to be harder than NP previous to this work. ... generalised games, very few examples ...

successful and highly flexible framework for modelling games as computations.

The core of this paper is the construction presented in Section IV: a universal Turing machine embedded into a game of *Magic: The Gathering*. As we can arrange for the victor of the game to be determined by the halting behaviour of the Turing machine, this construction establishes the following theorem:

Theorem 1: Determining the outcome of a game of *Magic: The Gathering* in which all remaining moves are forced is undecidable.

A. Previous Work

Prior to this work, no undecidable real games were known to exist. Demaine and Hearn (2009) [10] note that almost every real-world game is trivially decidable, as they produce game trees with only computable paths. They further note that Rengo Kriegspiel[1] is "a game humans play that is not obviously decidable; we are not aware of any other such game." It is conjectured by Auger and Teytaud (2012) [1] that Rengo Kriegspiel is in fact undecidable, and it is posed as an open problem to demonstrate any real game that is undecidable.

The approach of embedding a Turing machine inside a game directly is generally not considered to be feasible for real-world games [10]. Although some open-world sandbox games such as Minecraft and Dwarf Fortress can support the construction of Turing machines, those machines have no strategic relevance and those games are deliberately designed to support large-scale simulation. In contrast, leading formal theory of strategic games claims that the unbounded memory required to simulate a Turing machine entirely in a game would be a violation of the very nature of a game [9].

The computational complexity of *Magic: The Gathering* in has been studied previously by several authors. Our work is inspired by [4], in which it was shown that four-player *Magic* can simulate a Turing machine under certain assumptions about player behaviour. In that work, Churchill conjectures that these limitations can be removed and preliminary work ... lines is discussed in [5]. The computational ... of a particular decision ...

Video divulgativo sull'argomento

https://www.youtube.com/watch?v=pdmODVYPDLA