Rule-based Interactive Fiction

! * o- >_

Chris Martens * Zachary Sparks * Claire Alvis * Will Byrd

Carnegie Mellon * Indiana University

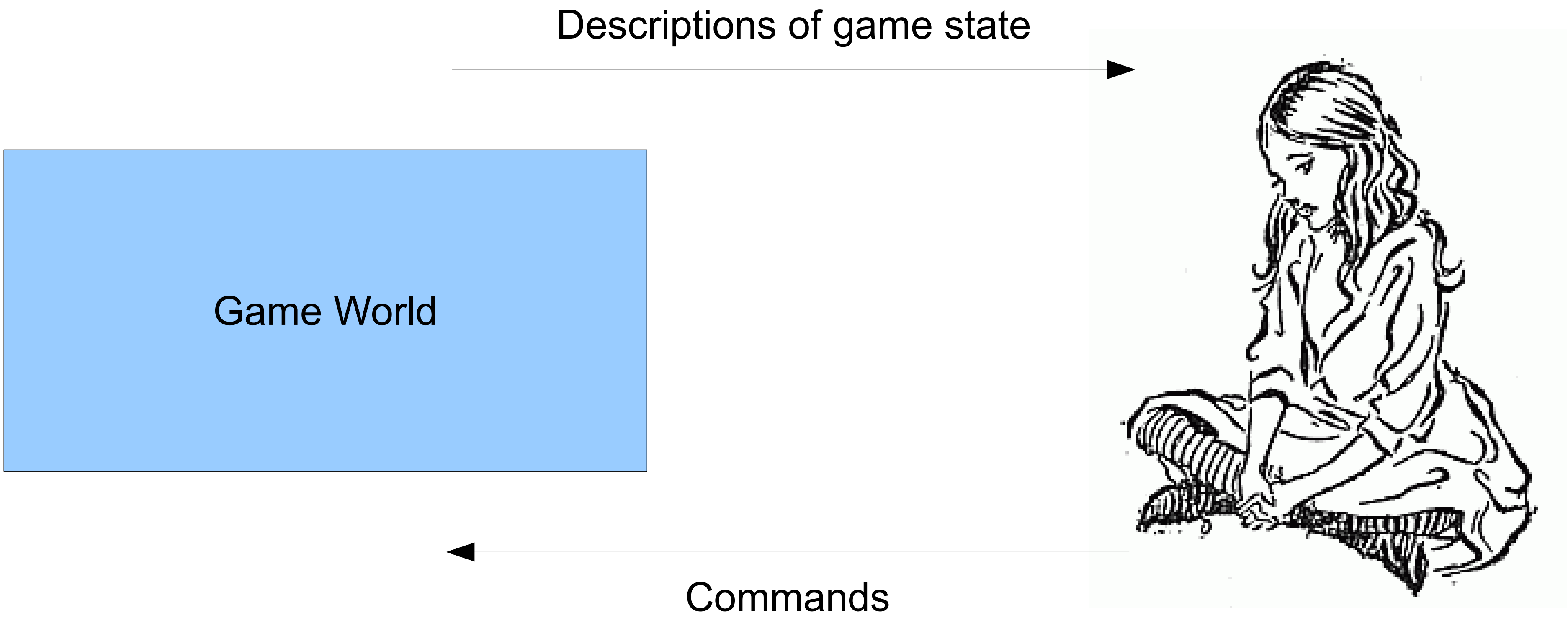
Intro to IF

```
West of House

You are standing in an open field west of a white house, with
a boarded front door.
There is a small mailbox here.

>_
```

Intro to IF



Descriptions of game state

Game World

Commands

Intro to IF

```
West of House

You are standing in an open field west of a white house, with
a boarded front door.
There is a small mailbox here.

>_
```

Intro to IF

```
> x house
The house is a beautiful colonial house which is painted white.
It is clear that the owners must have been extremely wealthy.
```

Intro to IF

```
> open mailbox
Opening the mailbox reveals a small leaflet.
```

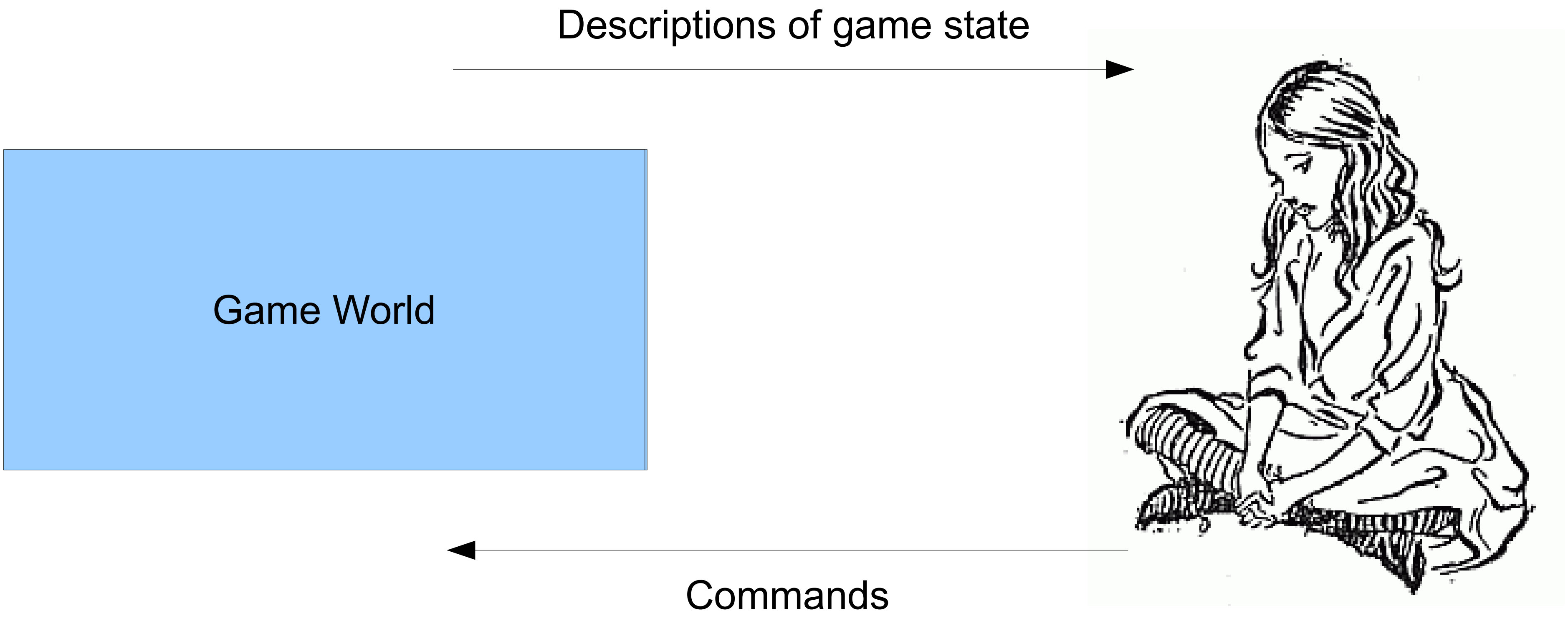
Intro to IF

```
> take leaflet
Taken.
```

Intro to IF

```
> inventory
You are carrying:
       A leaflet
```

Intro to IF

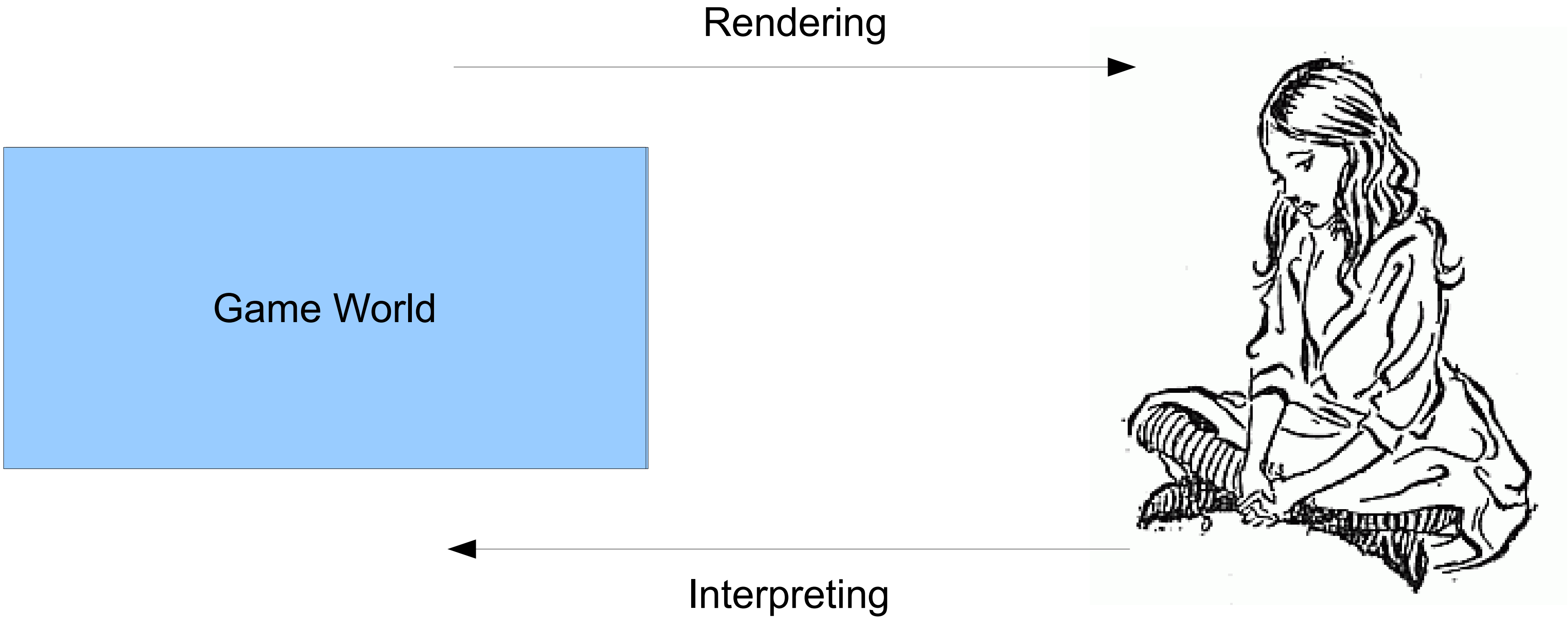


Descriptions of game state

Game World

Commands

Intro to IF



Rendering

Game World

Interpreting

A good domain for PL!

Really, this is a suggestion that we study all *interactive* programs in a declarative way.

IF is just fun :)

Logic Programming

Curry-Howard:

Props as Types / Proofs as Programs

Miller: ("Proof search foundations for logic programming," WOLLIC'03)

Props as Programs

Proof search as execution

Takeaway

Interactive proof search as *interactive* execution

(i.e. gameplay)

Takeaway

Interactive proof search as *interactive* execution

“[Building [proof] scripts is surprisingly addictive, in a videogame kind of way...”

Xavier Leroy
“Formal certification of a compiler back-end”
POPL'06

The Author's Task

- **Describe the world** (map, locations of objects, win conditions)
- **Describe the state transitions** that move the game forward
- **Anticipate player input**

Inform7 (see inform7.com)

```
There is a room called West of House. "You are standing in an open field
west of a white house, with a boarded front door."

The white house is a backdrop in West of House.
The description of the house is
"The house is a beautiful colonial house which is painted white. It is
clear that the owners must have been extremely wealthy."

The small mailbox is a container in West of House.
The small mailbox is closed and openable.
After opening the mailbox, say "Opening the small mailbox reveals a
leaflet."
Instead of taking the mailbox, say "It is securely anchored."

The leaflet is in the small mailbox.
The description of the leaflet is
"'WELCOME TO ZORK!'"
```

Inform7

```
There is a room called West of House.

The white house is a backdrop in West of House.

The small mailbox is a container in West of House.

The leaflet is in the small mailbox.
```

Inform7

```
There is a room called West of House. "You are standing in an open field
west of a white house, with a boarded front door."

The white house is a backdrop in West of House.
The description of the house is
"The house is a beautiful colonial house which is painted white. It is
clear that the owners must have been extremely wealthy."

The small mailbox is a container in West of House.
The small mailbox is closed and openable.
After opening the mailbox, say "Opening the small mailbox reveals a
leaflet."
Instead of taking the mailbox, say "It is securely anchored."

The leaflet is in the small mailbox.
The description of the leaflet is
"'WELCOME TO ZORK!'"
```

Logic Programming

```
> take leaflet
```

```
?- take(leaflet, X).
```

Logic Programming

```
> take leaflet
Taken.

?- take(leaflet, X).
X = say(“Taken.”)
```

Logic Programming

```
Try: X = say “taken”

    .                      .
    .                      .
    .                      .
visible(leaflet)  portable(leaflet)
___________________________________
take(leaflet, say “taken”).
```

Logic Programming

```
There is a room called West of House.
"You are standing..."
```

```
in(player, westofhouse).
description(westofhouse, "You are standing...").
```

Logic Programming

```
The small mailbox is a container in West of House.
The small mailbox is closed and openable.
```

```
in(mailbox, westofhouse).
closed(mailbox).
openable(mailbox).
```

Logic Programming

```
(Inform7 has defaults!)
```

```
examine(X, say(D))
  :- visible(X), description(X, D).
```

Logic Programming

```
open(C, say("opened"))
  :- openable(C), closed(C).
%% But also change state! Mailbox opened; contents visible...?
```

Logic Programming

```
take(X, say("taken"))
  :- portable(X), visible(X).
%% But also change in(X,Y) to in(X,player)!

%% And then there are all the failure conditions...
```

Linear Logic Programming

`*, -o, !`

A logic for reasoning about resources and state. * conjoins 2 resources, -o consumes a resource and produces another, ! recovers the original logic.

Linear Logic Programming

```
in(mailbox, westofhouse).
closed(mailbox).
!openable(mailbox).
```

Linear Logic Programming

```
open(C, say("opened")) * opened(C)
  o- !openable(C) * closed(C).

take(X, say("taken")) * in(X,player) * visible(X)
  o- !portable(X) * visible(X) * in(X,Y).
```

But there's a problem with this that we don't yet know how to solve...

Linear Logic Programming

```
take(X, say("taken"))
  * in(X,player)
  * visible(X)      %% ???
o- !portable(X)
  * visible(X)
  * in(X,Y).
```

It isn't clear whether to conserve this resource. How is it defined?

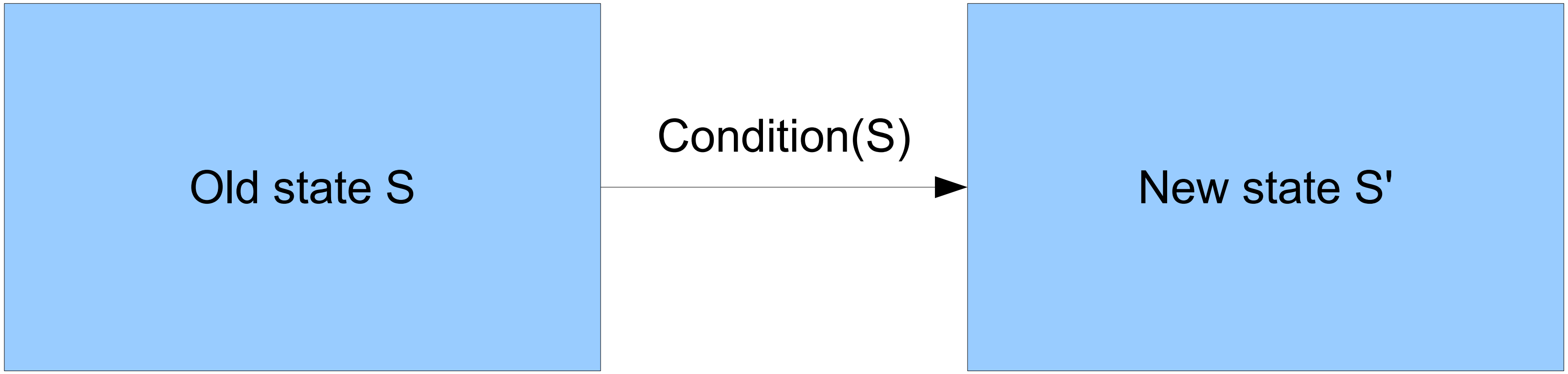
Linear Logic Programming

```
visible(X)
  o- in(player,R) * in(X,R).
visible(X)
  o- in(X,C) * open(C) * visible(C).
```

Linear Logic Programming

```
visible(X) * in(player,R) * in(X,R)
  o- in(player,R) * in(X,R).
visible(X)
    * in(X,C) * open(C) * visible(C)
  o- in(X,C) * open(C) * visible(C).
```

Linear Logic Programming



“Read-Only Access to Resources” a la Garg & Pfenning

Possibly Fruitful?

Proof Irrelevance

Hybrid Logic

Other Kripke-style modal logic

Other Challenges

Overriding Defaults

- It's visible *unless* the room is dark
- *unless* the player carries a flashlight
- *unless* the batteries are dead
- (and so on)

Plotkin, A. Rule-based programming. http://eblong.com/zarf/rule-language.html, June 2010.

Other Challenges

- Negation
  - Taking something: check whether the player already has it!

  - Could put the failure rule first...

Other Challenges

In general, giving the author control of rule precedence.

Possibly fruitful: *Defeasible Logic*

(Donald Nute, defeasible.org)

Summary

Interactive programs as interactive proof search!

Richer logics for hard problems!

Some systems to check out:

- Lolli: http://www.cs.cmu.edu/~fp/courses/15816-f01/software.html
- Lollibot/Ollibot: https://github.com/clf/ollibot