Thesis Proposal:
Logical Interactive Programming for Narrative Worlds

Chris Martens

December 6, 2013

My interest: supporting the design & analysis of game mechanics at a linguistic level.

Talk Outline

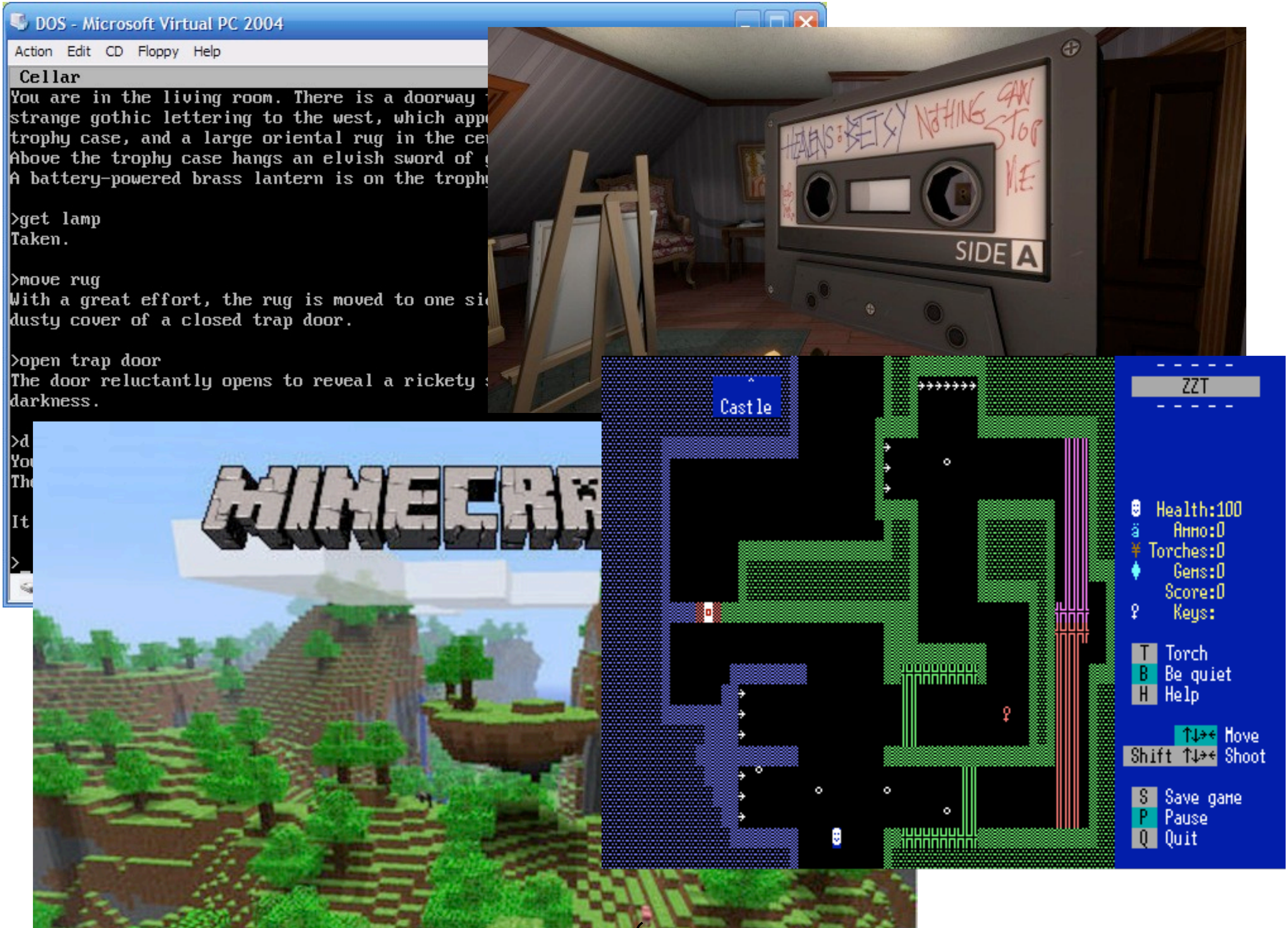
Section	Purpose
Narrative Worlds	define my target domain
Example: Blocks World	describe how CLF specification works
Supporting Interactivity and Analysis	describe my language extensions (phases, generative properties)
Narrative Worlds, revisited	give more examples to show breadth of scope
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

thesis statement

Talk Outline

Section	**Purpose**
Narrative Worlds	define my target domain
Example: Blocks World	describe how CLF specification works
Supporting Interactivity and Analysis	describe my language extensions (phases, generative properties)
Narrative Worlds, revisited	give more examples to show breadth of scope
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

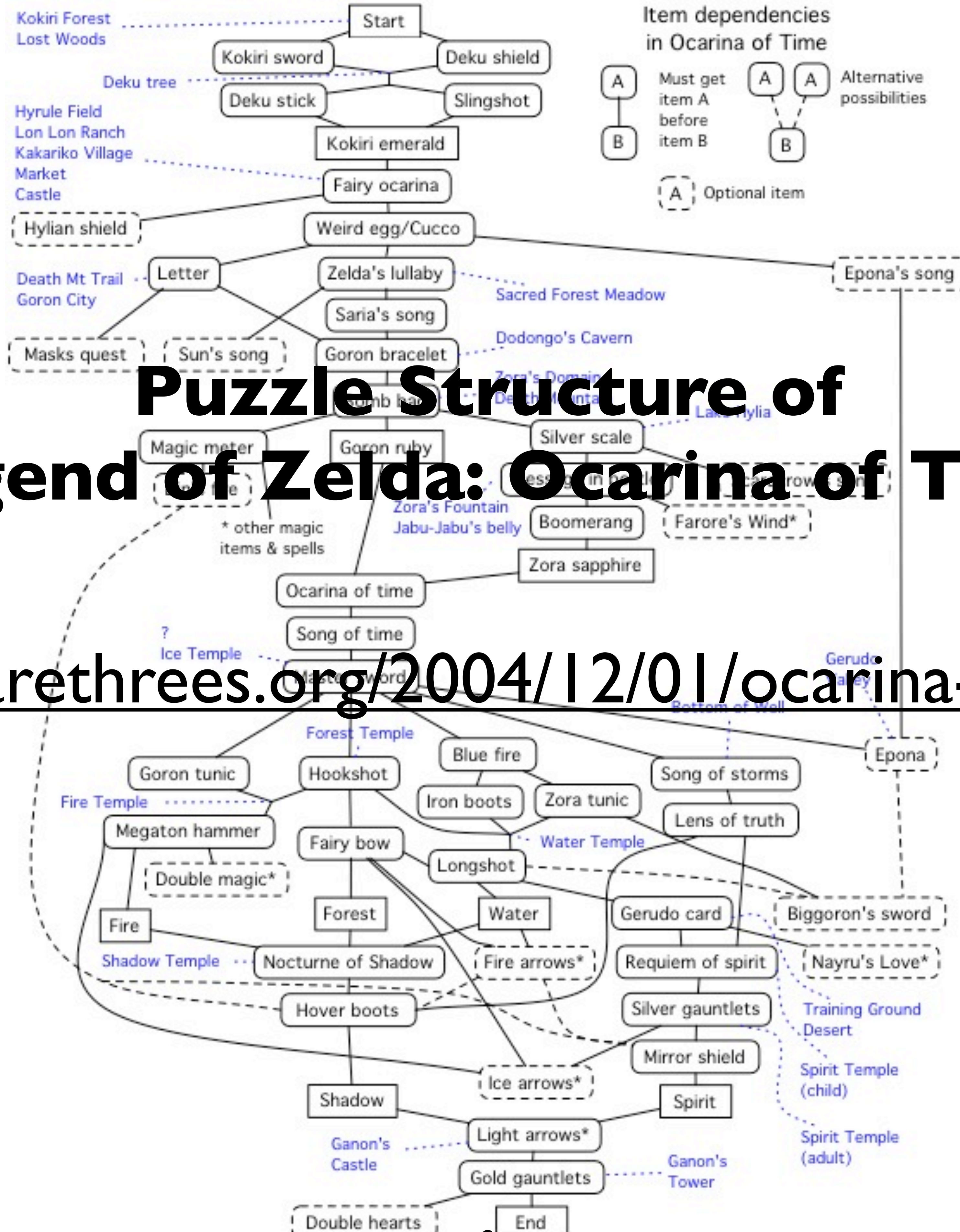
Narrative Worlds


DOS - Microsoft Virtual PC 2004
Action Edit CD Floppy Help
Cellar
You are in the living room. There is a doorway
strange gothic lettering to the west, which app
trophy case, and a large oriental rug in the ce
Above the trophy case hangs an elvish sword of
A battery-powered brass lantern is on the troph
>get lamp
Taken.
>move rug
With a great effort, the rug is moved to one si
dusty cover of a closed trap door.
>open trap door
The door reluctantly opens to reveal a rickety
darkness.
HEAVENS BETSY NOTHING CAN STOP ME
SIDE A
MINECRA
Castle
ZZT
Health:100
Ammo:0
Torches:0
Gems:0
Score:0
Keys:
T Torch
B Be quiet
H Help
Move
Shift Shoot
S Save game
P Pause
Q Quit

emphasis on narrative
“vs.”
emphasis on (open) worlds

“ludonarrative” = ludo (game/play) + narrative (story)

Puzzle Structure of Legend of Zelda: Ocarina of Time

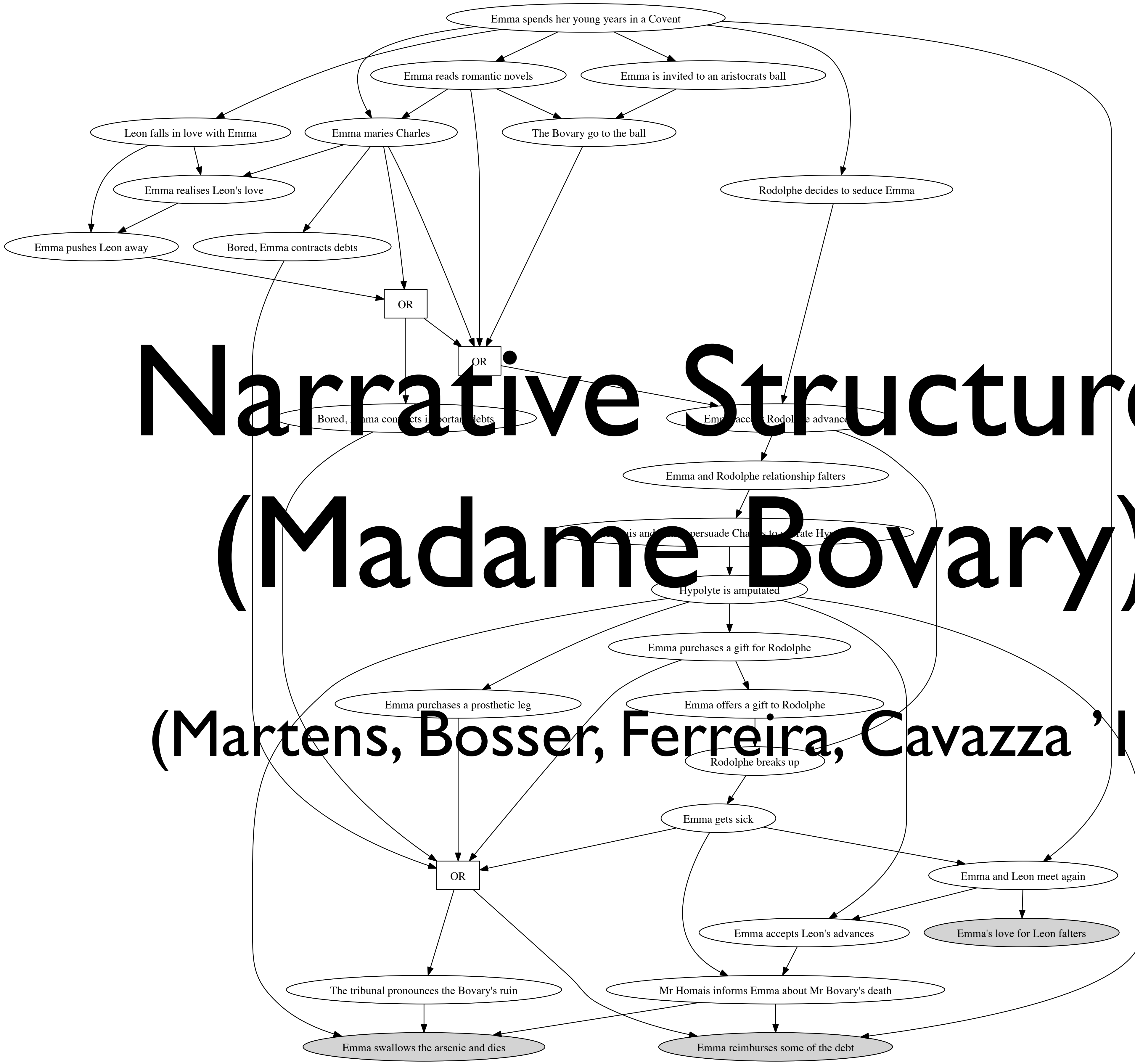


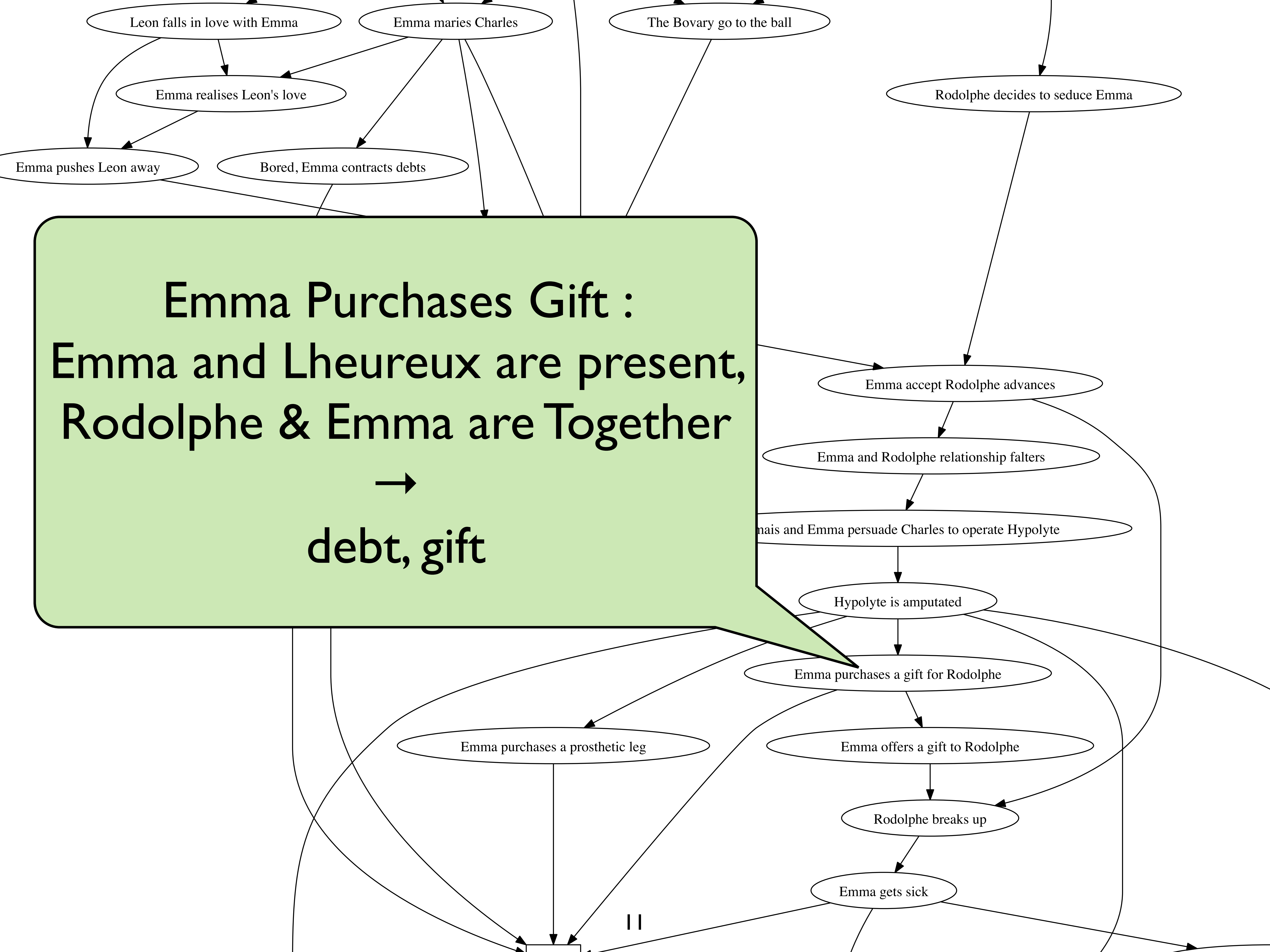
http://garethrees.org/2004/12/01/ocarina-of-time/

from Reddit /r/Minecraft

Narrative Structures (Madame Bovary)

(Martens, Bosser, Ferreira, Cavazza ’13)



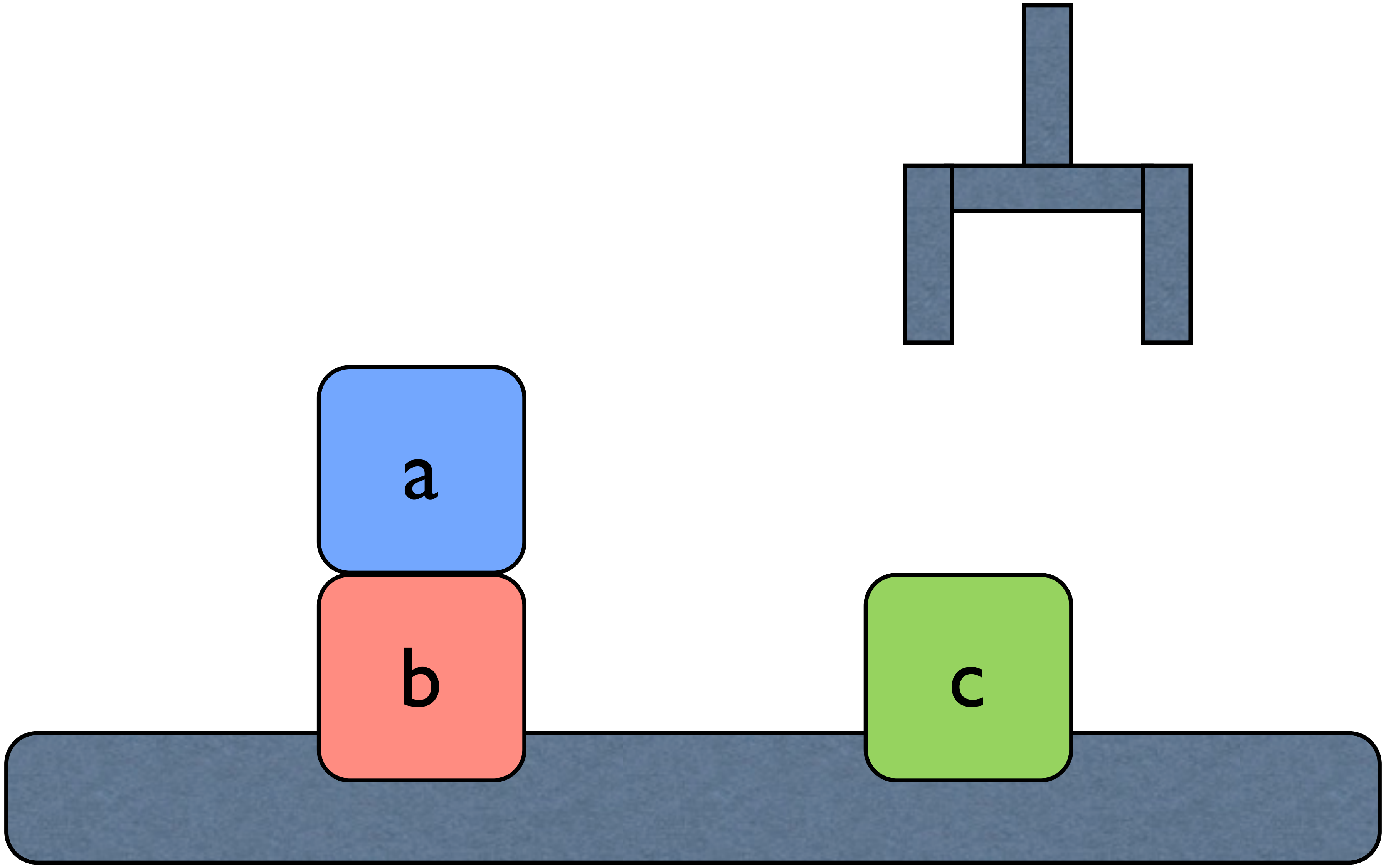

Leon falls in love with Emma
Emma maries Charles
The Bovary go to the ball
Emma realises Leon's love
Rodolphe decides to seduce Emma
Emma pushes Leon away
Bored, Emma contracts debts
Emma Purchases Gift :
Emma and Lheureux are present,
Rodolphe & Emma are Together
→
debt, gift
Emma accept Rodolphe advances
Emma and Rodolphe relationship falters
mais and Emma persuade Charles to operate Hypolyte
Hypolyte is amputated
Emma purchases a gift for Rodolphe
Emma purchases a prosthetic leg
Emma offers a gift to Rodolphe
Rodolphe breaks up
Emma gets sick

Shared structure: plots & puzzles create resource dependencies

Talk Outline

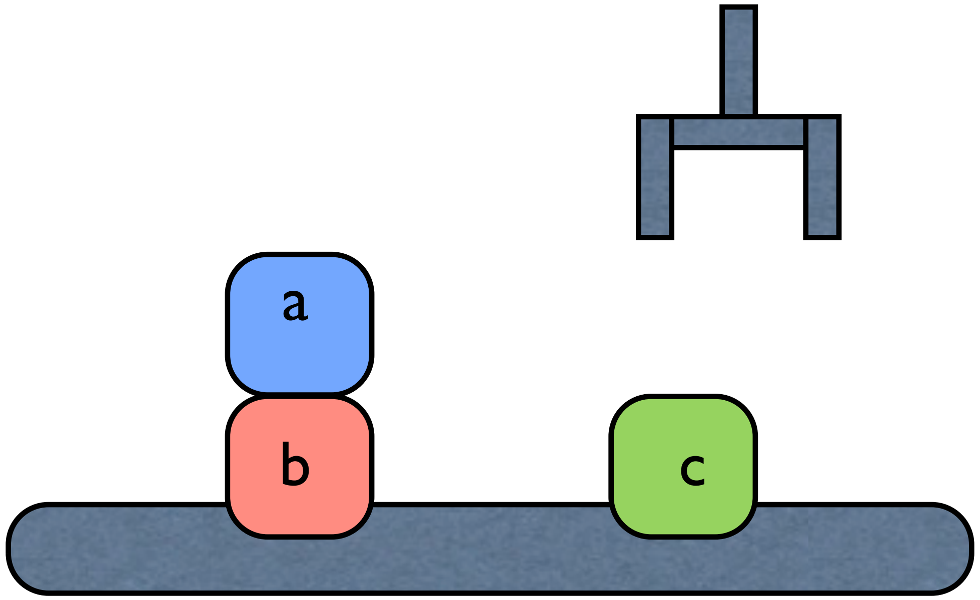
Section	**Purpose**
Narrative Worlds	define my target domain
Example: Blocks World	describe how CLF specification works
Supporting Interactivity and Analysis	describe my language extensions (phases, generative properties)
Narrative Worlds, revisited	give more examples to show breadth of scope
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

Simple Example: Blocks World



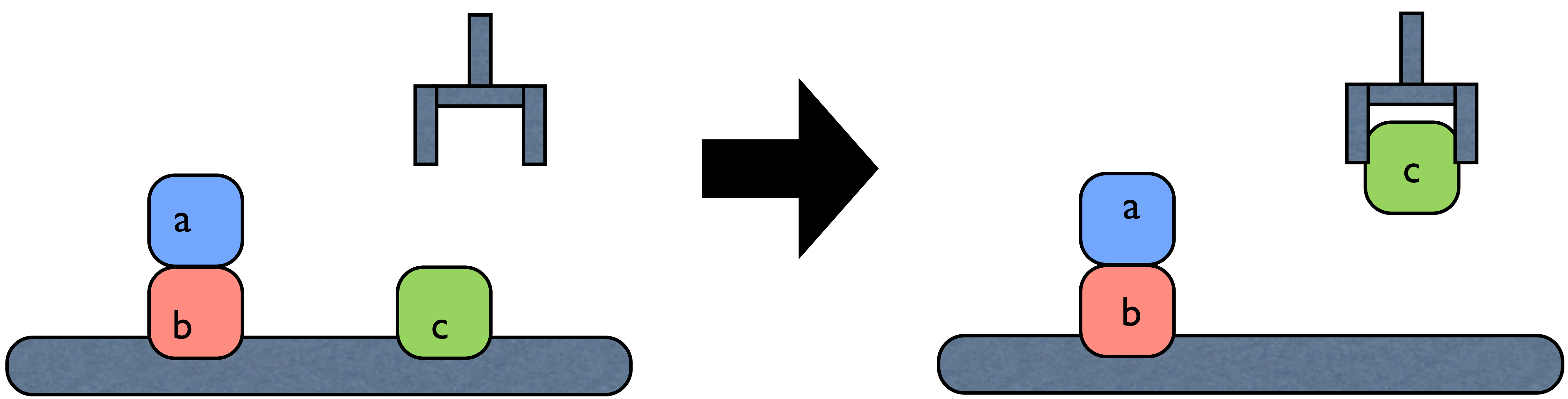
Representation of Individual States

{ arm_free,
on_table b,
on_table c,
clear c,
on a b,
clear a }



Representation of Action Rules

pickup_from_table :
on_table X * clear X * arm_free
-o {arm_holding X}.

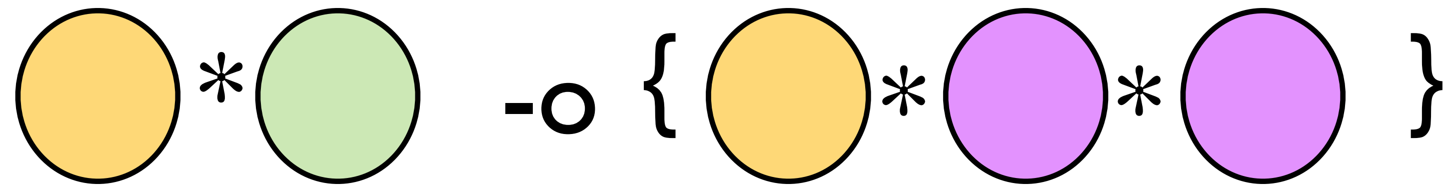


Blocks world cont'd

```
pickup_from_block :
  on X Y * clear X * arm_free
        -o {clear Y * arm_holding X}.

put_on_table :
  arm_holding X -o
       {on_table X * clear X * arm_free}.

put_on_block :
  arm_holding X * clear Y
     -o {on X Y * clear X * arm_free}.
```



Local state change

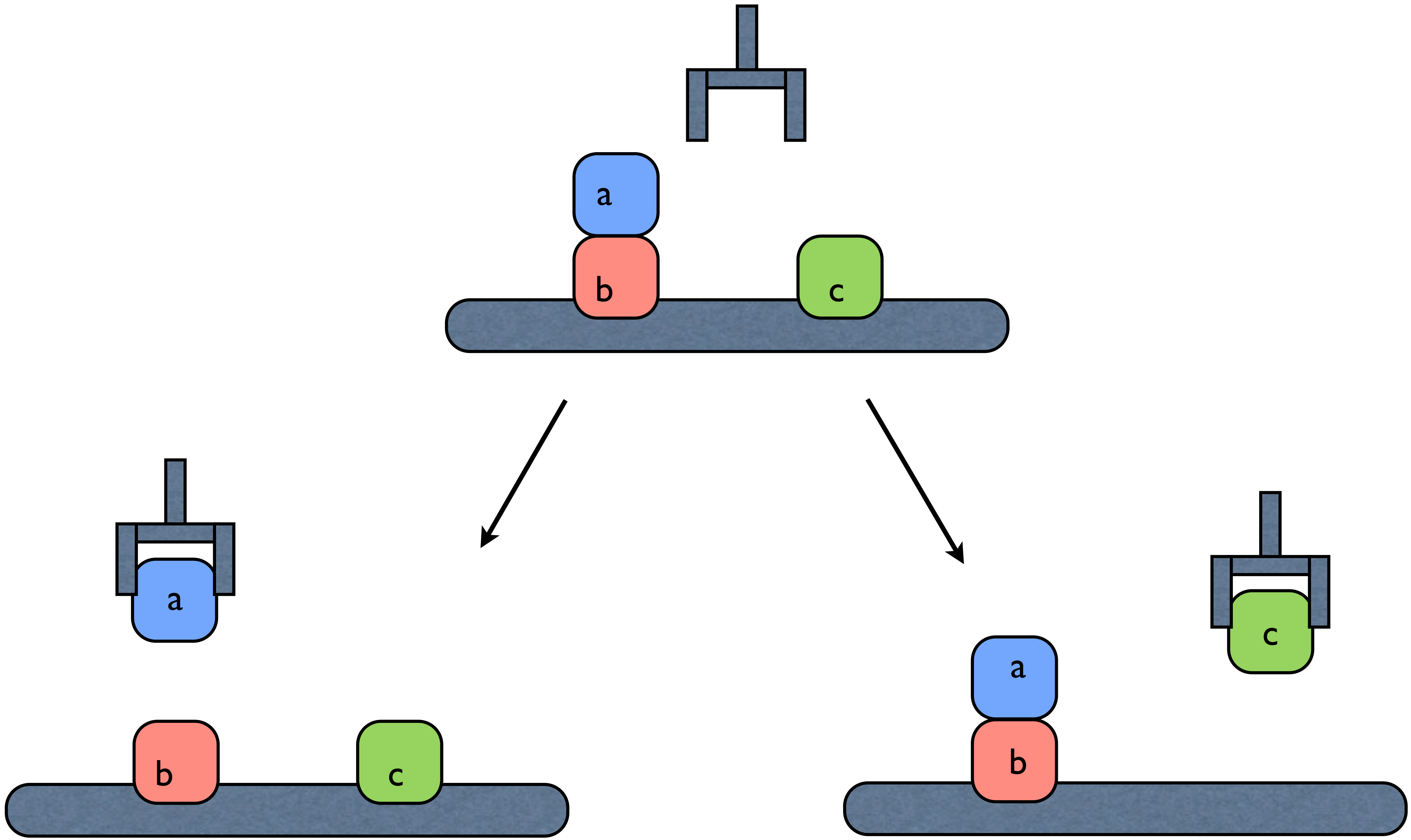
Celf Specification Framework

based on CLF (Watkins, Cervesato, Pfenning, Walker '02)

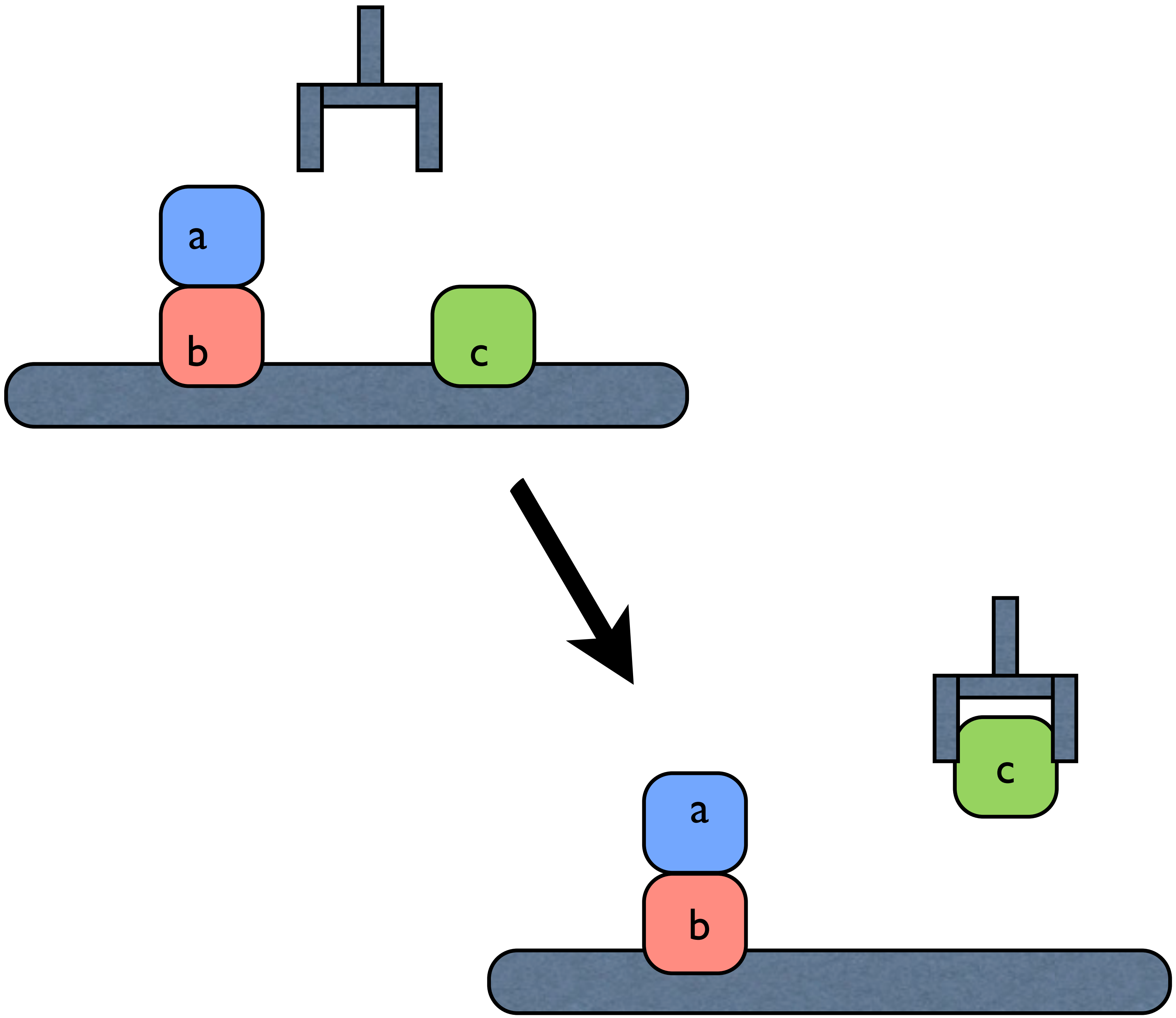
implements linear logic as a logic programming language (execution as proof search)

still many open questions about operational semantics

Committed Choice



Committed Choice

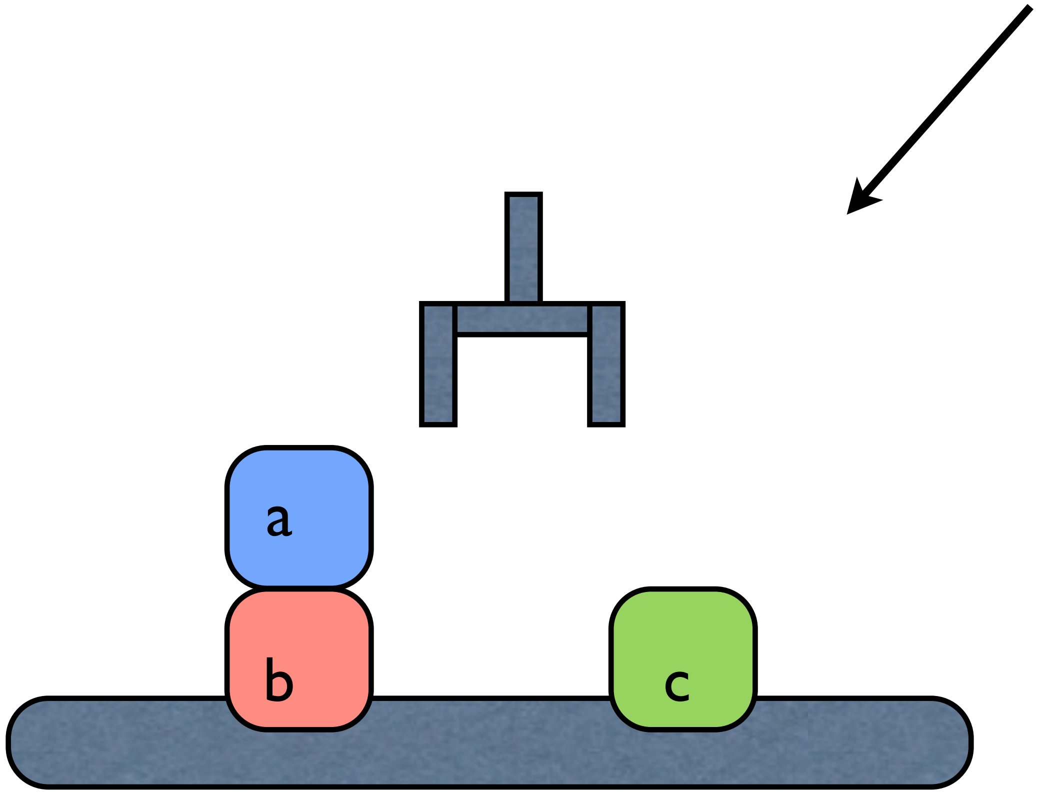


Celf

#query 10 (init -o {end_condition})

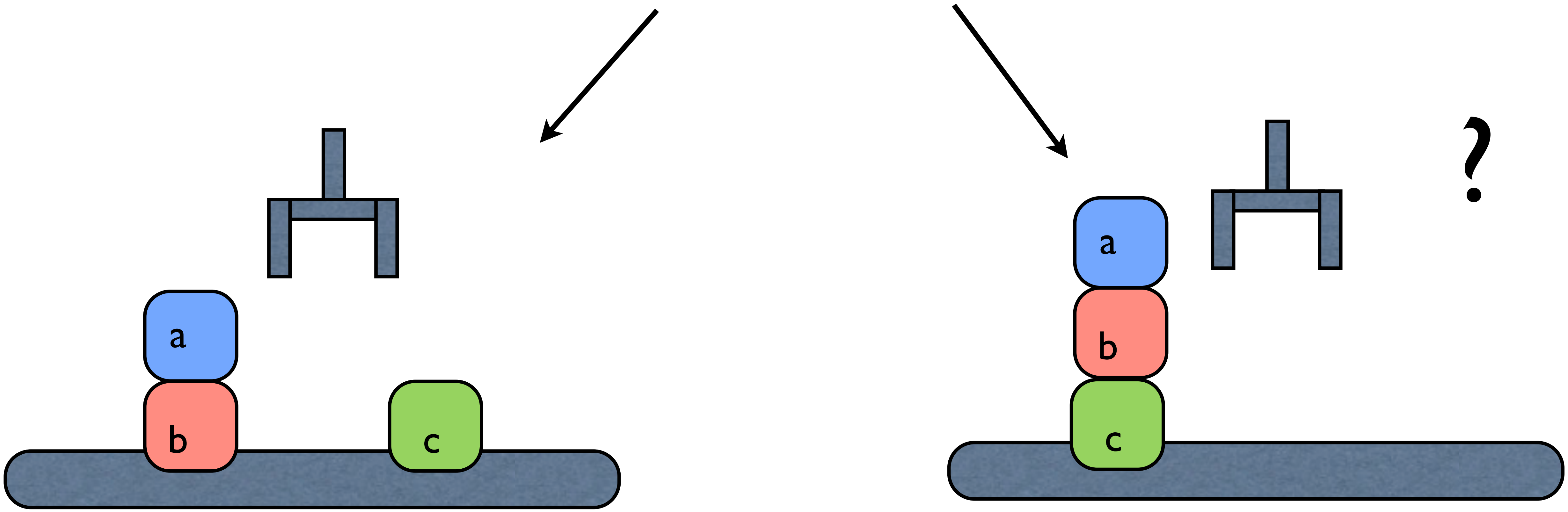
Celf

#query 10 (init -o {end_condition})



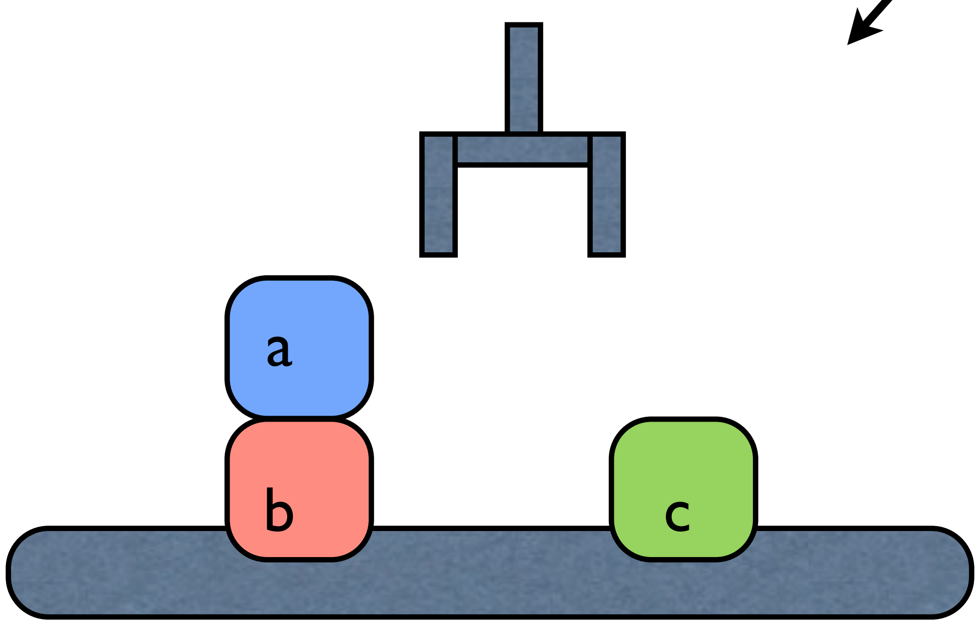
Celf

#query 10 (init -o {end_condition})



Celf

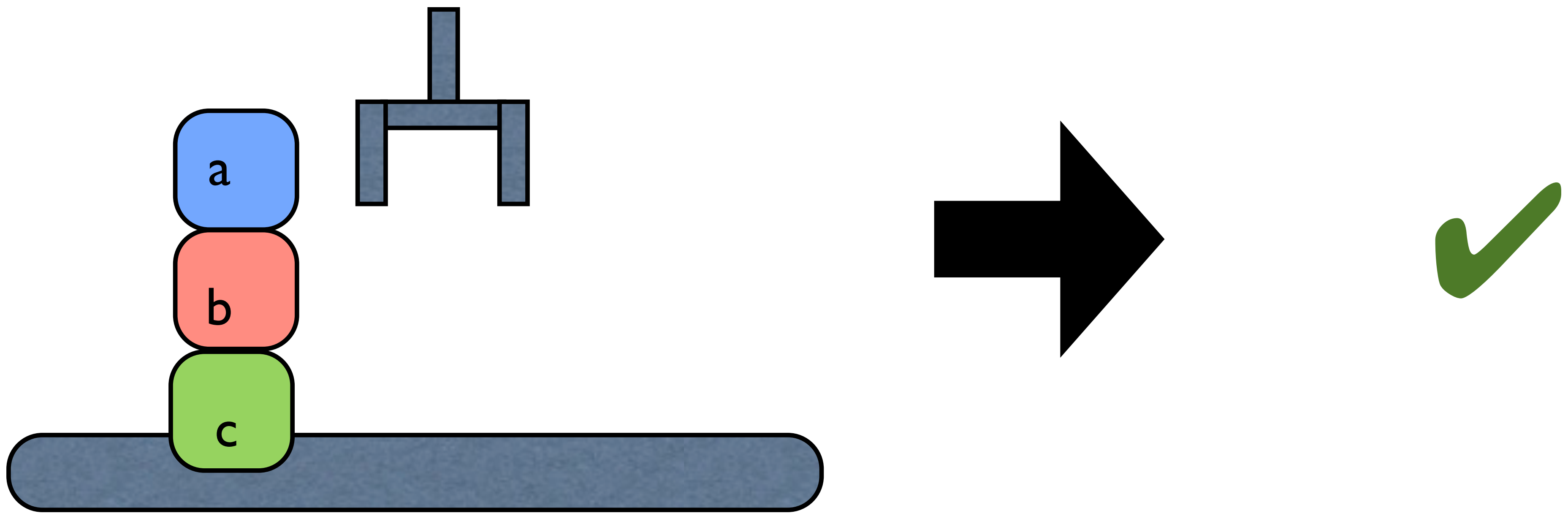
#query 10 (init -o {end_condition})



state at
quiescence

Celf

on a b * on b c * on_table c * arm_free
-o {**end_condition**}



Celf

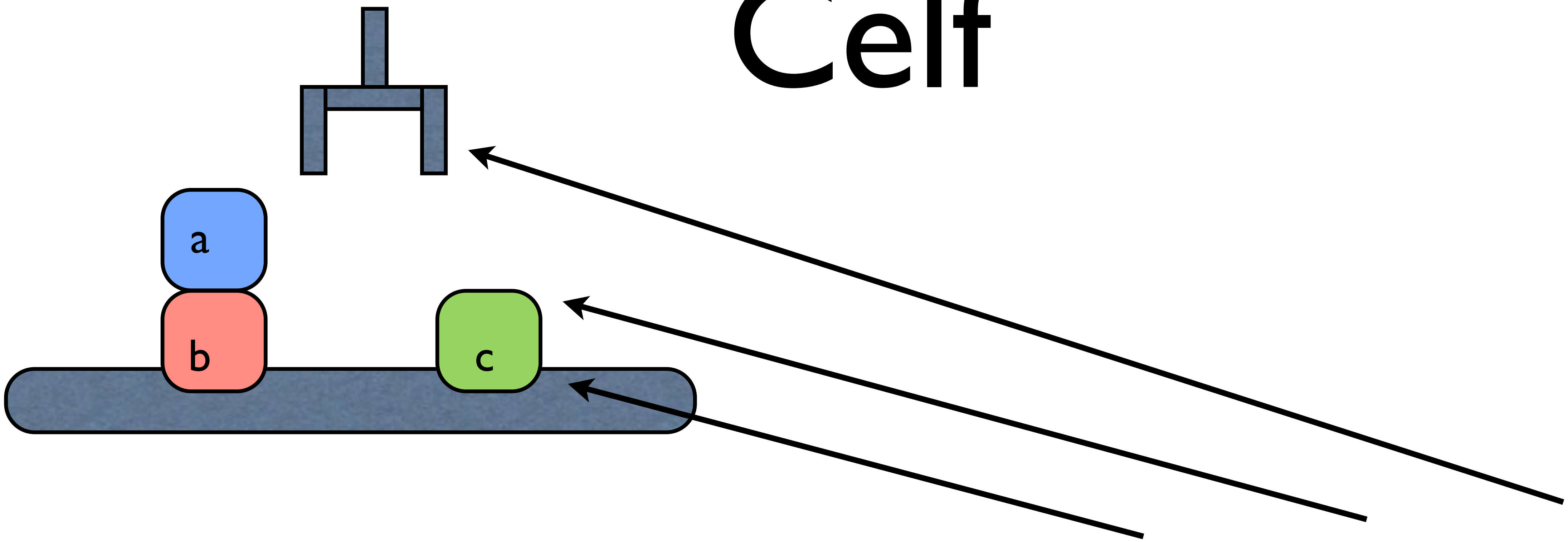
#query 10 (init -o {end_condition})

...

let {X13} = **pickup_from_table** [X10, [X11, X12]] in
let {[X14, [X15, X16]]} = **put_on_table** X13 in
let {X17} = **pickup_from_table** [X3, [X6, X16]] in
let {[X18, [X19, X20]]} = **put_on_block** [X17, X8] in

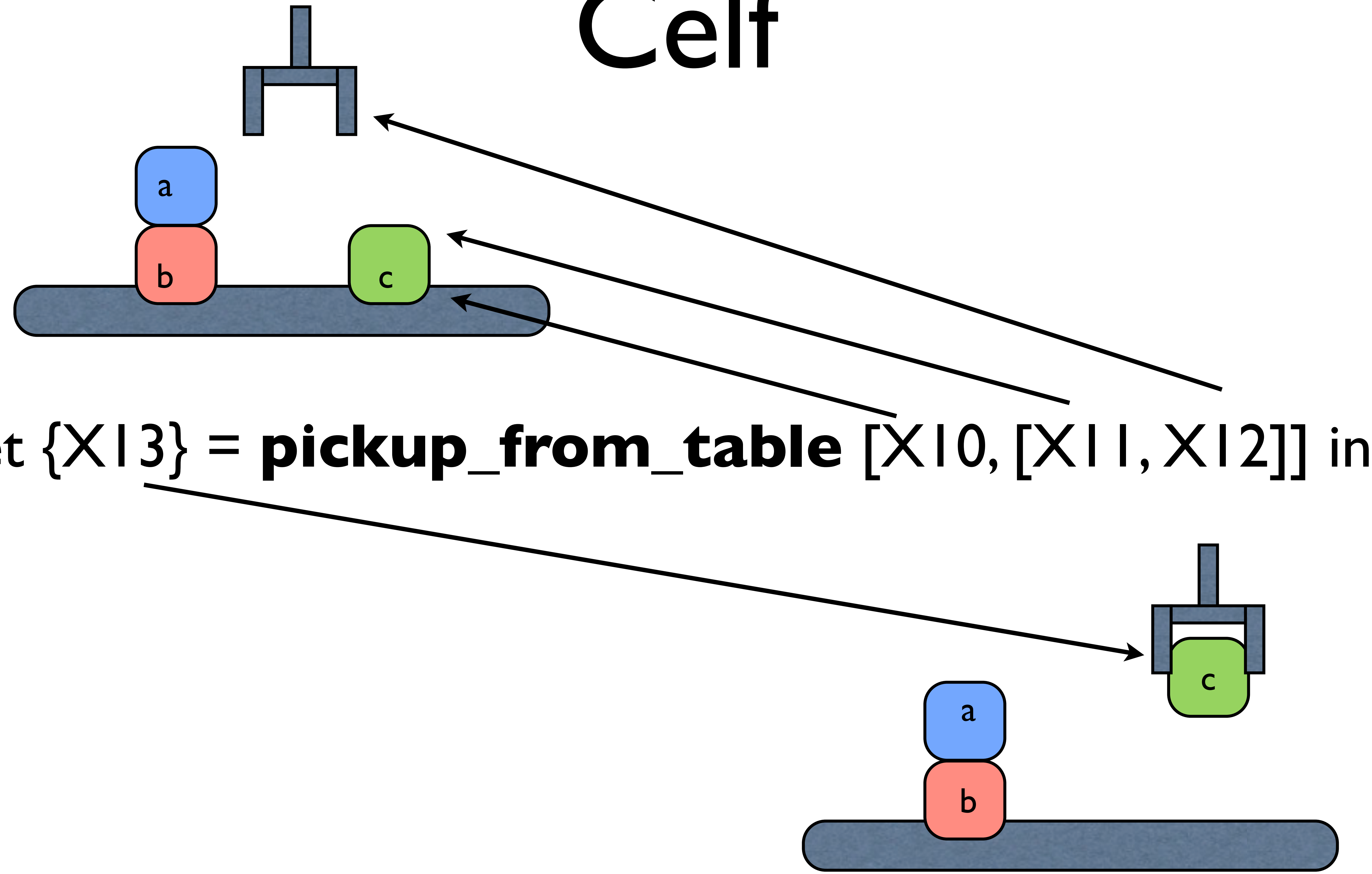
...

Celf



let {X13} = **pickup_from_table** [X10, [X11, X12]] in

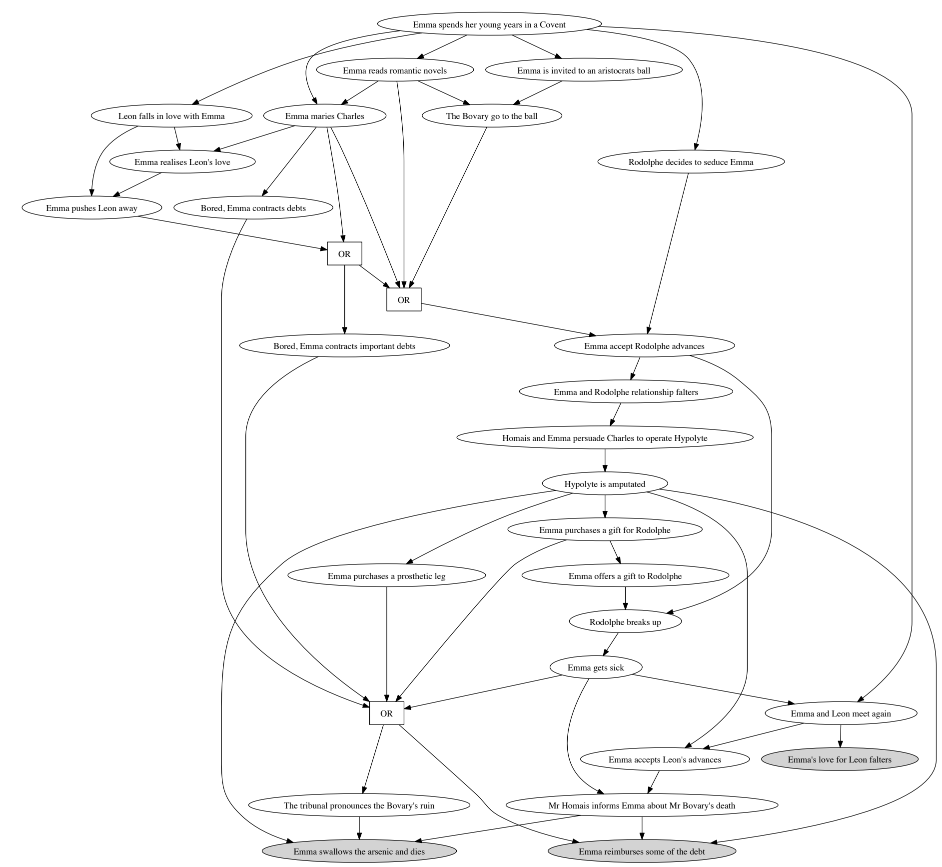
Celf



Celf

Proofs-as-traces: *structural artifacts* that we can analyze, e.g. for causal dependency.

...
let {X13} = **pickup_from_table** [X10, [X11, X12]] in
let {[X14, [X15, X16]]} = **put_on_table** X13 in
let {X17} = **pickup_from_table** [X3, [X6, X16]] in
let {[X18, [X19, X20]]} = **put_on_block** [X17, X8] in
...



Celf

Proofs-as-traces:
structural artifacts that we can analyze, e.g. for causal dependency.

c.f: PlotEx
http://eblong.com/zarf/plotex/

GraphPlan
http://www.cs.cmu.edu/~avrim/graphplan.html

Proto-Thesis Statement

Linear logic programming can form the basis of a framework for specifying simulation mechanics.

Proto-Thesis Statement

Linear logic programming can form the basis of a framework for [specifying]+ [simulation]+ mechanics.

Proto-Thesis Statement

[Linear logic programming]+ can form the basis of a framework for [specifying]+ [simulation]+ mechanics.

Talk Outline

Section	Purpose
Narrative Worlds	define my target domain
Example: Blocks World	describe how CLF specification works
Supporting Interactivity and Analysis	describe my language extensions (phases, generative properties)
Narrative Worlds, revisited	give more examples to show breadth of scope
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

Adding interactivity to blocks world

action : type.
pickup : block → action.
putdown_on : block → action.
putdown_table : action.
stop : action.

Interactivity cont'd

```
pickup_from_block :
  current (pickup X)
  * on X Y * clear X * arm_free
            -o {clear Y * arm_holding X}.
```

Interactivity cont'd

Where does "**current**" come from?
The engine & player should "take turns."

current (pickup X) * … -o {… * **player_turn**}
current (putdown_table X) * … -o {… * **player_turn**}
…

player_turn -o {ForAny a:action. **current** a}

Phases

Block-delimited subsignatures

```
phase world = {
  rule1  : current Action * … -o {…}.
  rule2 : current Action * … -o {…}.
}

phase player = {
   rule : player_turn -o {…}
}
```

Phases

Connected by specification of *quiescence* behavior

```
phase world = {...}

phase player = {...}

quiesced world -o
  {player_turn * phase player}.

quiesced player -o {phase world}.
```

Phases

Connected by specification of *quiescence* behavior

```
phase world = {...}

phase player = {...}

quiesced world -o
  {player_turn * phase player}.

quiesced player -o {phase world}.
```

Related: "sensing" and "action" atoms in Meld (Claytronics)

Phases

…are block-delimited subsignatures connected by specifications of quiescence behavior.

quiesced P * *State* -o {**phase** P’ * *State’*}.

arbitrarily many phases
looping + branching

Compiling Phases

We can interpret phase-structured programs as programs in Celf.

Compiling Phases

We can interpret phase-structured programs as programs with higher-order, mixed-chaining rules in Celf.

Compiling Phases

We can interpret phase-structured programs as programs with higher-order, mixed-chaining rules in Celf.

(see proposal document for details)

Compiling Phases

We can interpret phase-structured programs as programs with higher-order, mixed-chaining rules in Celf.

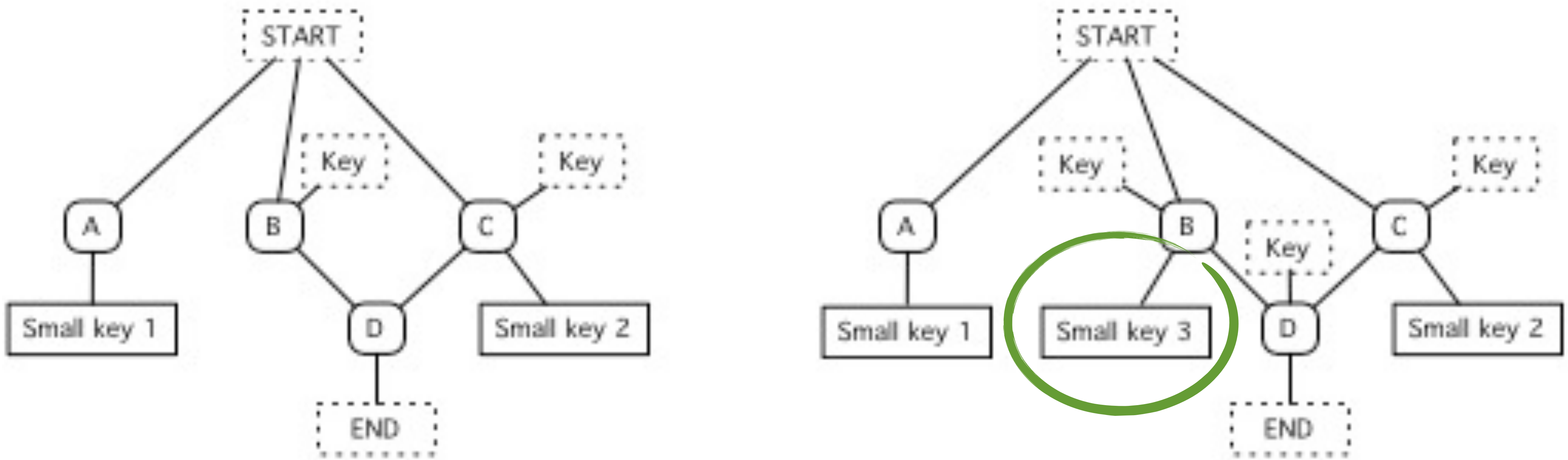
Ongoing work: Check that the source-level semantics corresponds to compiled semantics.

Thesis Statement

Phase-structured *linear logic programming can form the basis of a framework for specifying,* ***testing, and inventing ludonarrative mechanics****.*

Checked Metatheory

http://garethrees.org/2004/12/01/ocarina-of-time/



Dungeon A: may become unsolvable

Dungeon B: always solvable

Blocks World

http://www.cs.cf.ac.uk/Dave/AI2/node116.html

If the arm is holding a block, it is not empty.

If block A is on the table it is not on any other block.

If block A is on block B, block B is not clear.

Generative Properties

a way of stating and checking programmer intent

Generative Properties

based on Generative Invariants (Simmons ’12)

Generative Invariants

To prove an invariant of a signature Σ:

Describe a signature Σ_{gen} with a distinguished start state (usually an atom "gen")

and prove that

- initial states of Σ are in (could be generated by) Σ_{gen}
- every rule in Σ preserves membership in Σ_{gen}

Generative Invariants

```
gen          -o {genArm * !genBlocks}.
genArm       -o {arm_free}.
genArm       -o {arm_holding X}.

genBlocks -o {on_table X * genTop X}.
genBlocks * genTop Y
             -o {on X Y * genTop X}.

genTop X -o {clear X}.
```

Quiescence & Activity

Quiescence: no rules can fire

Activity: at least one rule can fire

Activity Generator for Blocks World

act -o {arm_holding X * !actBlocks}.
act -o {arm_free * clear Y * !actBlocks}.
actBlocks -o {on_table X}.
actBlocks -o {on X Y}.
actBlocks -o {clear X}.

Ongoing Work:

Work out how to mechanically check these properties.

Show applicability to invariant properties of game worlds.

Talk Outline

Section	**Purpose**
Narrative Worlds	define my target domain
Example: Blocks World	describe how CLF specification works
Supporting Interactivity and Analysis	describe my language extensions (phases, generative properties)
Narrative Worlds, revisited	give more examples to show breadth of scope
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

Narrative Worlds, Revisited

Generalized Narrative Structures

As a player, you get to ***select a character****,* ***guide their choices****,* ***watch other characters react*** *to what you've chosen, and* ***accomplish (or fail at) your chosen goals****.*

Generalized Narrative Structures

do/murder :
anger C C' * anger C C' * anger C C' * anger C C'
* at C L * at C' L * has C weapon -o
{at C L * has C weapon * !dead C' * !murdered C C'}.

do/thinkVengefully :
loves C C’ * !murdered K C’
-o {loves C C’ * anger C K * anger C K}.

Generalized Narrative Structures

do/murder : **do C (murder C’) ***
anger C C' * anger C C' * anger C C' * anger C C'
* at C L * at C' L * has C weapon -o
{at C L * has C weapon * !dead C' * !murdered C C'}.

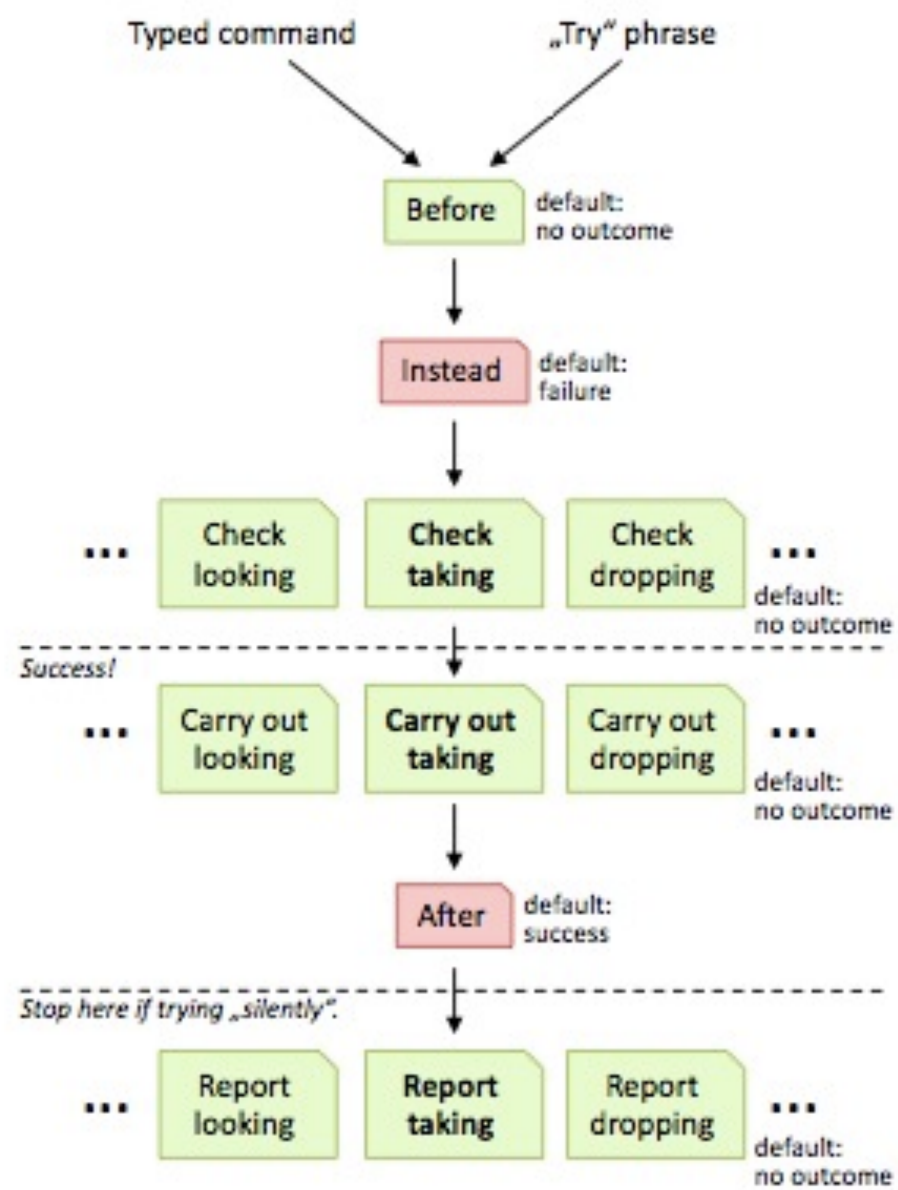
do/thinkVengefully : **do C (thinkVenge K) ***
loves C C’ * !murdered K C’
-o {loves C C’ * anger C K * anger C K}.

Generalized Narrative Structures

Ongoing work: figure out how to specify failure conditions when preconditions for an action are not met.

Generalized Narrative Structures

Inform 7 action processing:



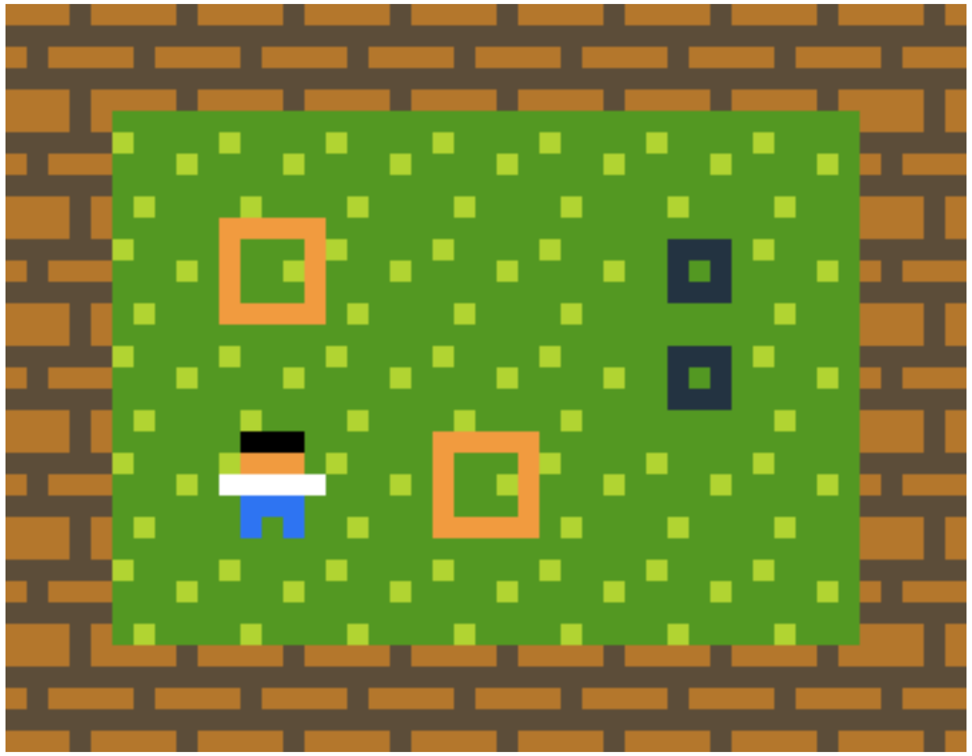
Implementable as phases!

Puzzle games

Scope: PuzzleScript

http://www.puzzlescript.net

Sokoban

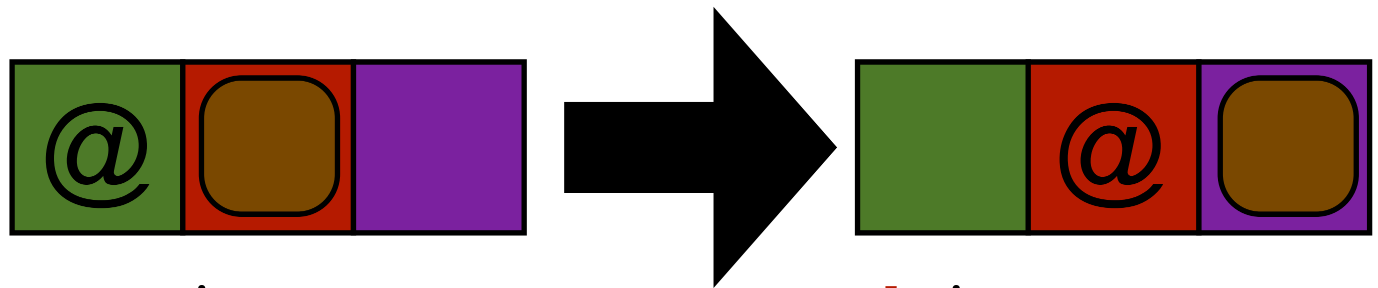


In PuzzleScript:

[> Player | Crate] -> [> Player | > Crate]

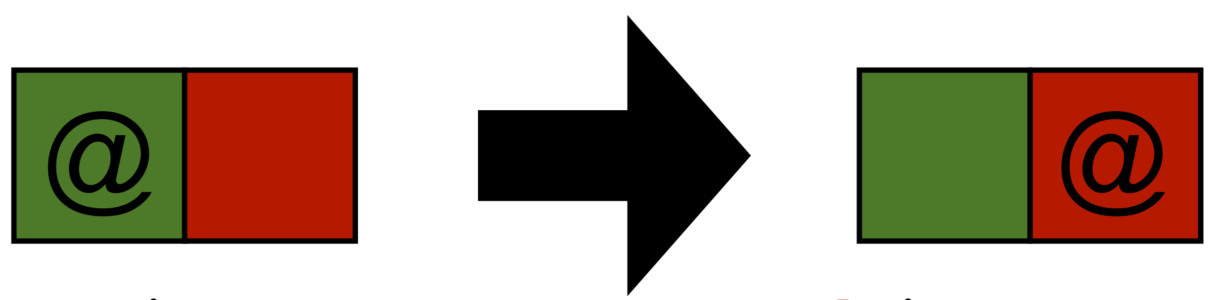
Sokoban Rules

push :



loc pusher L * in_dir L Dir L' * loc block L'
 * in_dir L' Dir L'' * empty L''
 -o {empty L * loc pusher L' * loc block L''}.

move :



loc pusher L * in_dir L Dir L' * empty L'
 -o {empty L * loc pusher L'}.

```
push :
action (arrow Dir) *
loc pusher L * in_dir L Dir L' * loc block L'
          * in_dir L' Dir L'' * empty L''
  -o {empty L * loc pusher L' * loc block L''}.

move :
action (arrow Dir) *
loc pusher L * in_dir L Dir L' * empty L'
  -o {empty L * loc pusher L'}.
```

Many more examples (some in progress):

https://github.com/chrisamaphone/interactive-lp/tree/master/examples

Talk Outline

Section	**Purpose**
Narrative Worlds	define my target domain
Example: Blocks World	step through all the pieces of my proposal
Narrative Worlds, revisited	show the intended scope of those ideas
Proposed Work & Evaluation Strategy	establish a plan to justify my thesis statement

Proposed Work

Shortcoming of Existing Framework	Proposed Solution
Sometimes we want to impose partial orderings among rules.	Language proposal with phases.
Programming with state is hard to reason about!	**Machine-checked** invariants and other characterizations of states; analysis tools such as causality and dependency graphs.
Non-interactive, low-feedback programming workflow.	**Visual state editor and trace rendering.**
Lack of access to common game programming libraries for e.g. graphical rendering, text parsing, etc.	**Implement compatibility between the language and existing game frameworks** (e.g. Twine)

Evaluation

Phase-structured linear logic programming can form the basis of a framework for specifying, testing, and inventing ludonarrative mechanics.

How will I determine success?

Phase-structured linear logic programming can form the basis of a framework for specifying, testing, and inventing ludonarrative mechanics.

Develop several examples in the framework.

Phase-structured linear logic programming can form the basis of a framework for specifying, testing, and inventing ludonarrative mechanics.

Prove correspondence and build prototype.

Phase-structured linear logic programming can form the basis of a framework for specifying, testing, and inventing ludonarrative mechanics.

Design UI & tooling, including visual rendering.

Phase-structured linear logic programming can form the basis of a framework for specifying, testing, and inventing ludonarrative mechanics.

Generative properties and graphical analysis tools

Key Contributions

To game design:

- simple, uniform logical formalism
- executable specs ("sketching" systems)
- reasoning and intent-checking tools as an integrated part of design process

Key Contributions

To logical frameworks:

- exploration of a new domain as evidence for its generality
- new or alternative answers to open questions about semantics
- establishment of metatheoretic tools

Timeline

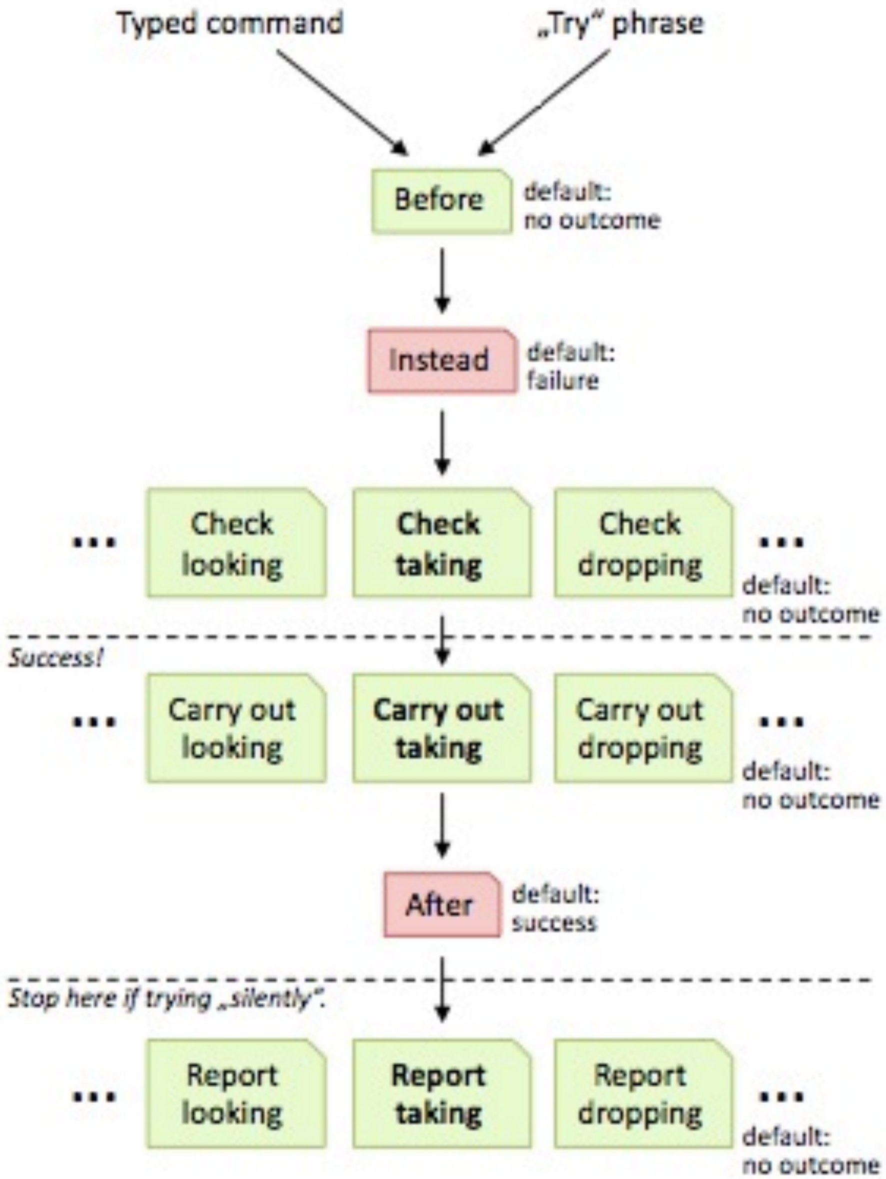
- **Spring 2014:** Finish working out theoretical concerns (language semantics, proofs, and sketch of generative property checking)
- **Summer-Fall 2014:** Implementation of prototype and development of examples
- **Spring 2015:** Write dissertation
- **Summer 2015:** Defend dissertation

Thank You!

extra slides

Phase links for Inform7 action processing graph:

```
qui read -o {phase parse}.
qui parse * outcome none
  -o {message defaultParseError * phase report}
qui parse * outcome failure -o {phase report}.
qui parse * outcome success -o {phase check1}.
qui check1 -o {phase check2}.
qui check2 * outcome success
  -o {phase carryout}.
qui check2 * outcome failure -o {phase report}.
qui check2 * outcome none
  -o {message default * phase report}.
qui carryout -o {phase report}.
```



Phases Inform7 action processing:

```
phase check1 = {
  - : init * outcome X -o {outcome none}.

  - : $action (take Obj) * inventory Obj
      -o {outcome failure
         * message "You already have it."}.

  - : $action look -o {outcome success}.
}
phase check2 = {
  - : $action (take Obj) * outcome none
        * visible Obj -o {outcome success}.
}


phase carryout = {
  - : action (take Obj) * in Obj C
      -o {inventory Obj * message "taken"}.
  - : action look * $in player R * $description R D
      -o {message D}.
}
```