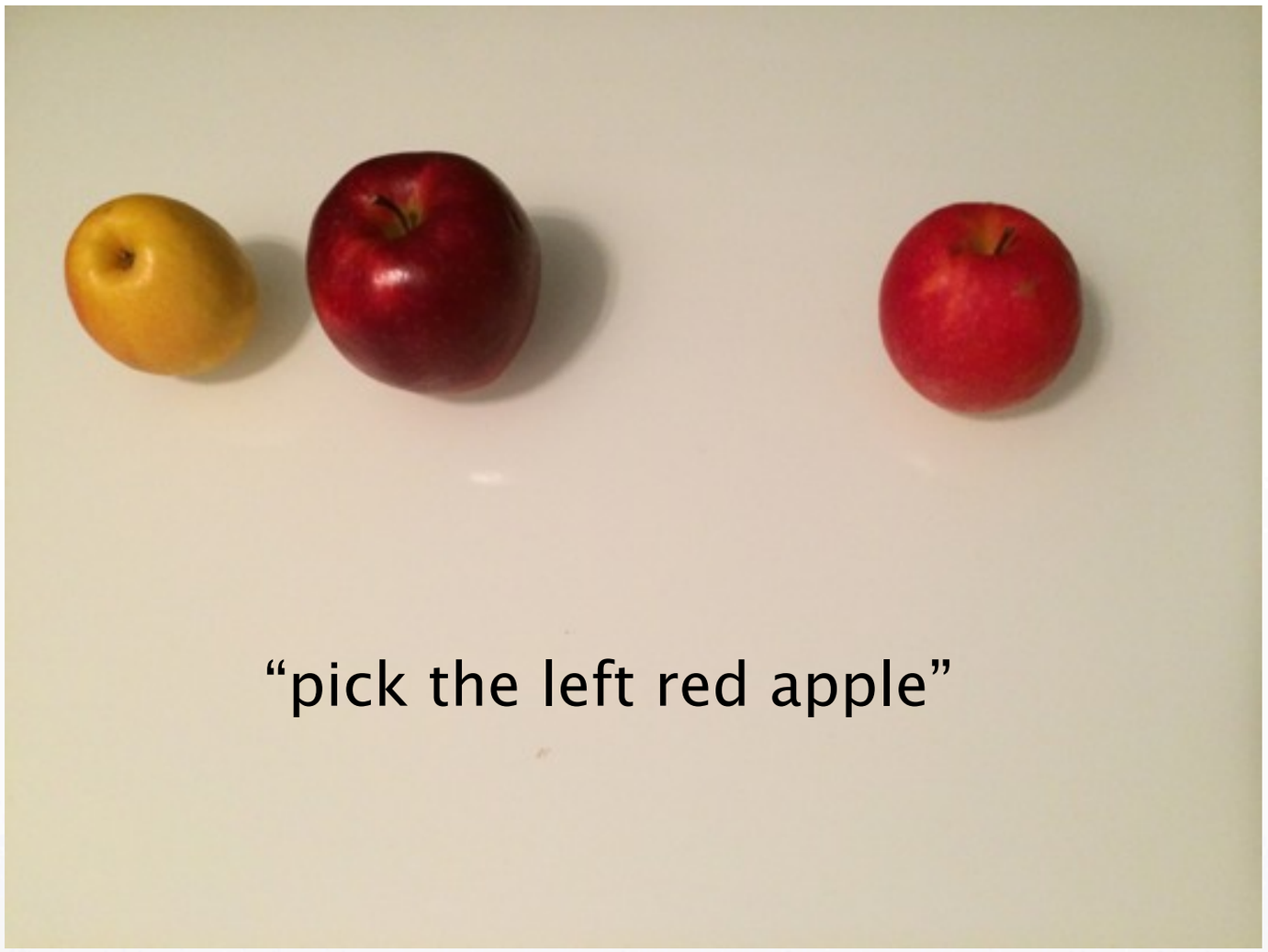
Listener gaze: Understanding and reacting to generated instructions

Nikolina Koleva

Saarland University
Department of Computational Linguistics
January 13, 2014

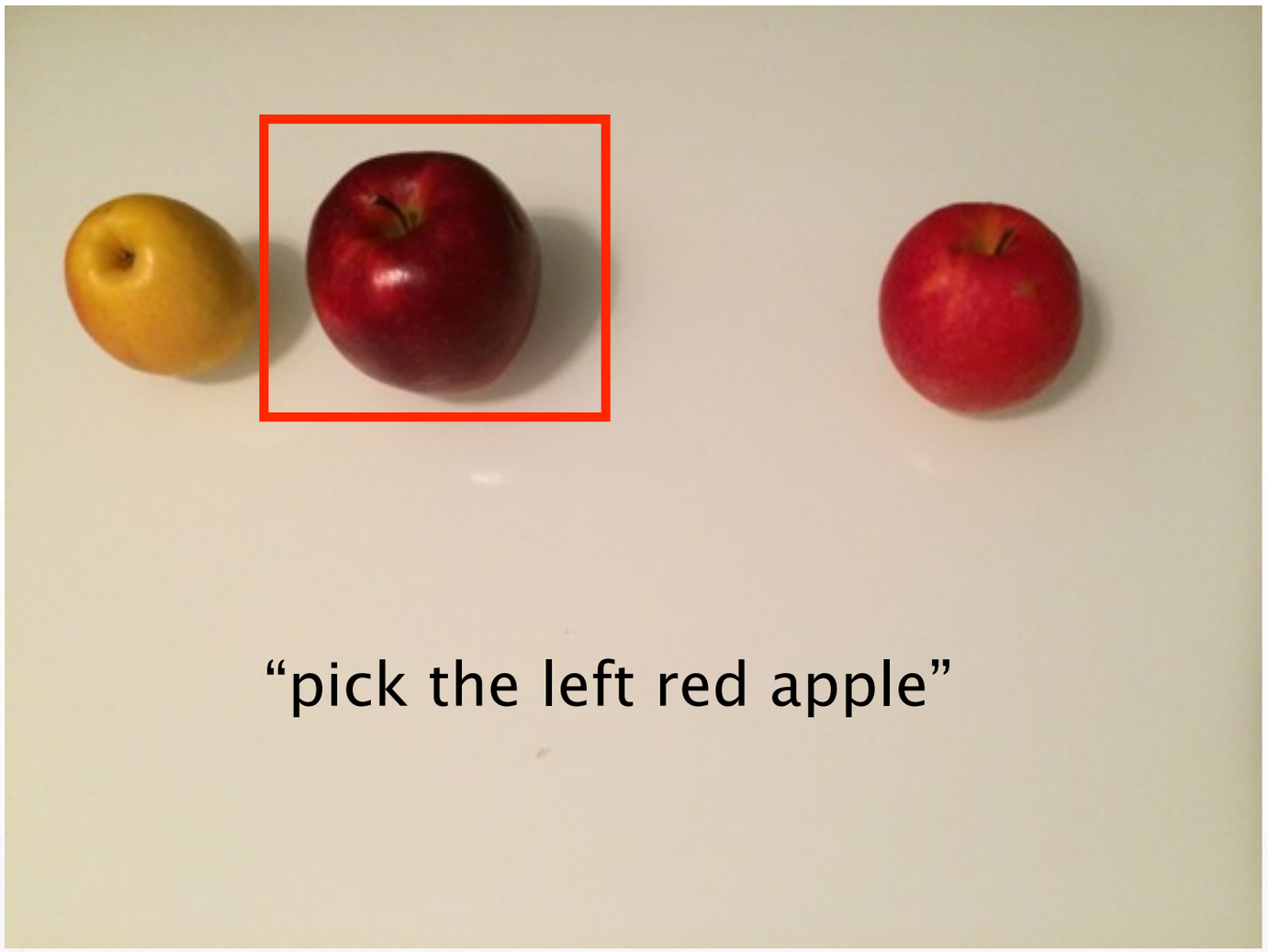
Motivation



"pick the left red apple"

- not the leftmost apple
- not the only red apple

Motivation



- not the leftmost apple
- not the only red apple

Outline

1. Motivation
2. Background
3. Automated Planning for situated NLG (Garoufi and Koller 2010)
   - Sentence Generation as Planning
   - Taking non-linguistic context into account
4. Enhancing Referential Success by Tracking Hearer Gaze (Koller et al. 2012)
   - Interactive NLG in virtual environments
   - Experimental Set up and Evaluation

Background

Natural Language Generation (NLG)

- tables
- web sites
- images
- videos
- ...

- summary
- caption
- instruction
- ...

Referential Expressions

- **Pronouns**: he, she etc.
- **Definite Noun Phrases**: the window, the car etc.
- **Spatial and Temporal Reference**: to the left, tonight etc.

Tree Adjoining Grammar

- mildly context-sensitive formalism
- more powerful than CFGs
- captures global and local dependencies
- basic elements of **TAG**: elementary trees
  - ➡ at least one lexical item associated with a tree: anchor
- derivation by applying substitution and adjunction

Automated Planning

Planning Task

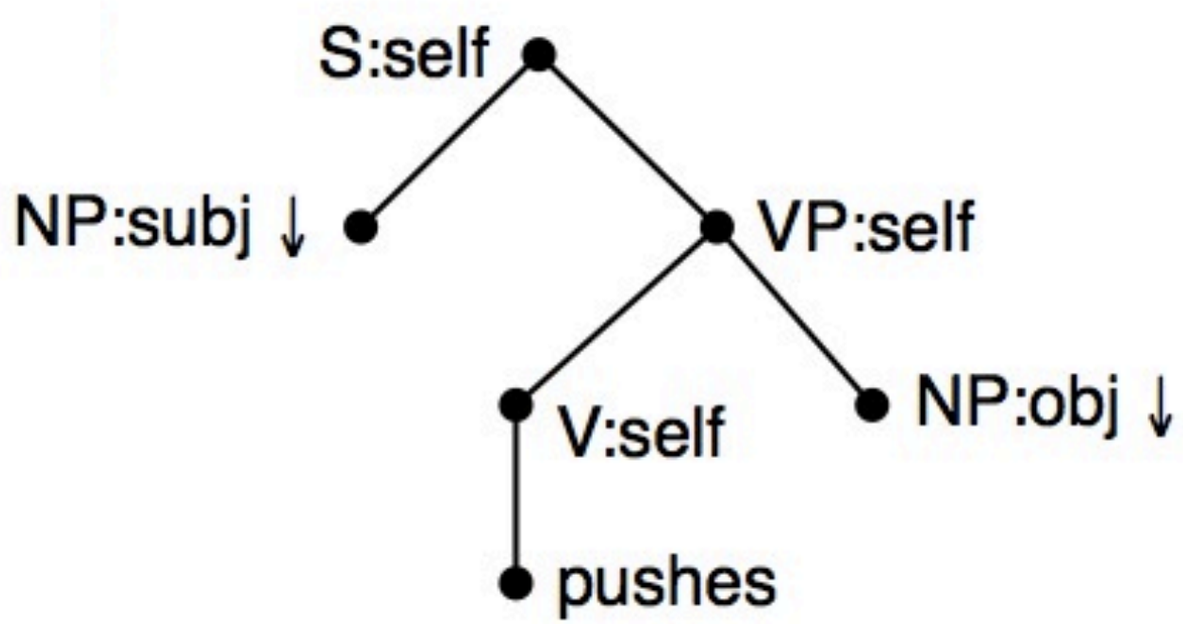
- Given
  - **state variables**
  - **actions as preconditions and effects**
  - **initial state**
  - **goal state**
- Compute sequence of actions that leads from the **initial** to a **goal** state

Automated Planning for situated NLG (Garoufi and Koller 2010)

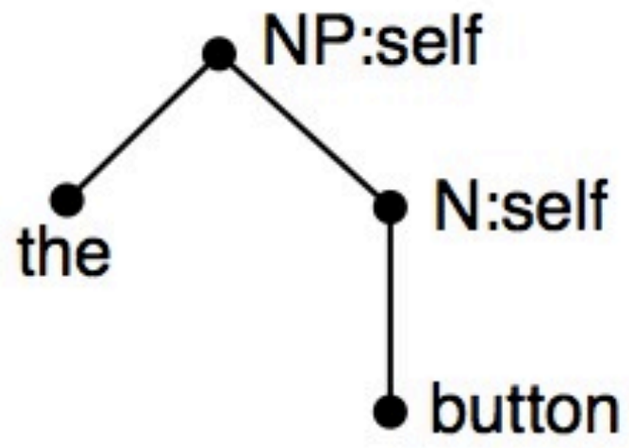
NLG as Planning

- **Input**
  - grammar
  - knowledge base (KB)
  - comunicative goal
- **Task**
  - compute a derivation of a sentence for the comunicative goal
- **Constraints**
  - grammaticality
  - completeness
  - all REs resolvable

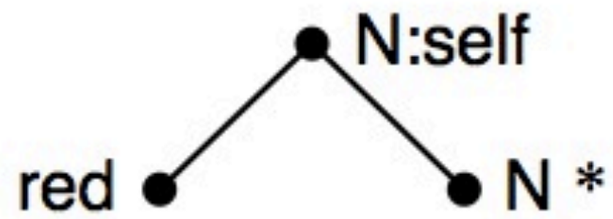
TAG Example



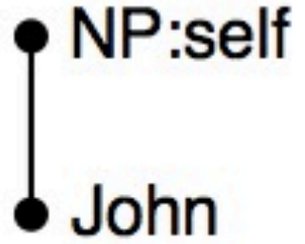
semcontent: {push(self,subj,obj)}



semcontent: {button(self)}



semcontent: {red(self)}



semcontent: {John(self)}

TAG Example

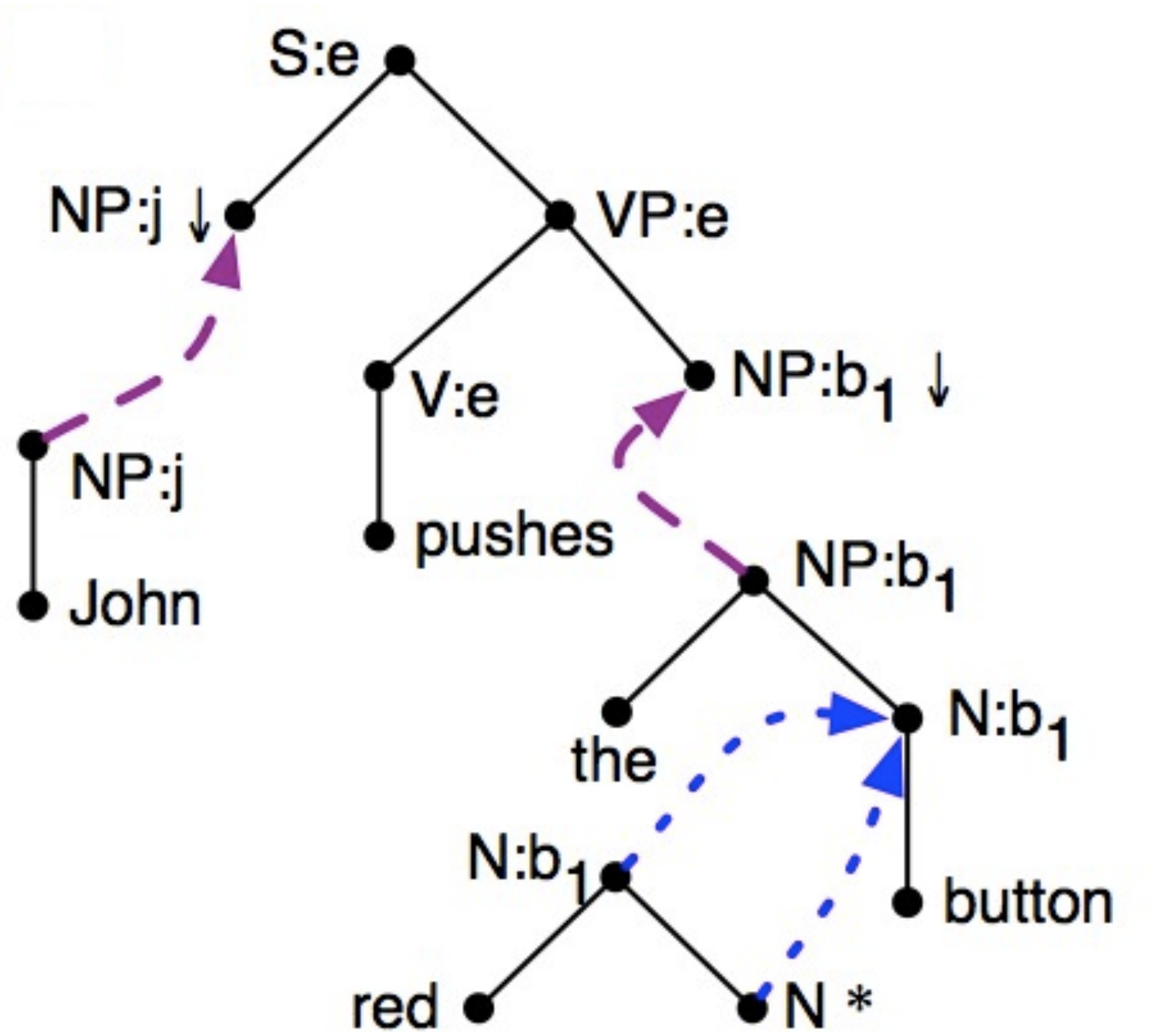
KB:

{push(e, j, b1),
John(j), button(b1),
button(b2), red(b1)}

TAG Example

KB:

{push(e, j, b1),
John(j), button(b1),
button(b2), red(b1)}



Elementary Trees as Operators

pushes(root, n1, n2, n3, e, j, b1):

Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1),
next(n1, n2), next(n2, n3)

Effect: ¬subst(S,root),subst(NP, n1),subst(NP, n2), ref(n1, j), ref(n2, b1),
∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):

Precond: subst(NP, n1), ref(n1, j), John(j)

Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):

Precond: subst(NP, n2), ref(n2, b1), button(b1)

Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):

Precond: canadjoin(N, n2), ref(n2, b1), red(b1)

Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Derivation as Plan

Initial state:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), subst(S, root), ref(root, e)}

Goal state:

{∀A ∀u. ¬subst(A, u), ∀u ∀x. ¬distractor(u, x)}

Derivation as Plan

Initial state:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), subst(S, root), ref(root, e)}

Goal state:

{∀A ∀u. ¬subst(A, u), ∀u ∀x. ¬distractor(u, x)}

grammatical completeness

Derivation as Plan

Initial state:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), subst(S, root), ref(root, e)}

Goal state:

{∀A ∀u. ¬subst(A, u), ∀u ∀x. ¬distractor(u, x)}

grammatical completeness

unique references

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), subst(S, root), ref(root, e)}

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), subst(S, root), ref(root, e)}

---

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root),subst(NP, n1),subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Match Preconditions

State:

{**push(e, j, b1)**, John(j), button(b1), button(b2), red(b1), **current(n1)**, **next(n1, n2)**, **next(n2, n3)**, **subst(S, root)**, **ref(root, e)**}

pushes(root, n1, n2, n3, e, j, b1):
Precond: **subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)**
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Apply Effects

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ~~subst(S, root)~~, ref(root, e), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), distractor(n1, e), distractor(n1, b1), distractor(n1, b2), distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: **¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)**

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Match Preconditions

State:

{push(e, j, b1), **John(j)**, button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), **subst(NP, n1)**, subst(NP, n2), **ref(n1, j)**, ref(n2, b1), distractor(n1, e), distractor(n1, b1), distractor(n1, b2), distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: **subst(NP, n1), ref(n1, j), John(j)**
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2 y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Apply Effects

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), ~~subst(NP, n1)~~, subst(NP, n2), ref(n1, j), ref(n2, b1), ~~distractor(n1, e)~~, ~~distractor(n1, b1)~~, ~~distractor(n1, b2)~~, distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: **¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)**

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), subst(NP, n2), ref(n1, j), ref(n2, b1, distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), subst(NP, n2), ref(n1, j), ref(n2, b1, distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

Which is the next action ?

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Match Preconditions

State:

{push(e, j, b1), John(j), **button(b1)**, button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), **subst(NP, n2)**, ref(n1, j), **ref(n2, b1)**, distractor(n2, e), distractor(n2, b1), distractor(n2, b2)}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: **subst(NP, n2), ref(n2, b1), button(b1)**
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Apply Effects

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), ~~subst(NP, n2)~~, ref(n1, j), ref(n2, b1), ~~distractor(n2, e)~~, distractor(n2, b1), ~~distractor(n2, b2)~~, **canadjoin(N, n2)**}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: **¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)**

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), ref(n1, j), ref(n2, b1), distractor(n2, b1), canadjoin(N, n2)}

Goal state:

{∀A ∀u. ¬subst(A, u), ∀u ∀x. ¬distractor(u, x)}

Derivation as Plan

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), ref(n1, j), ref(n2, b1), distractor(n2, b1), canadjoin(N, n2)}

Is that a goal state?

Goal state:

{∀A ∀u. ¬subst(A, u), ∀u ∀x. ¬distractor(u, x)}

Match Preconditions

State:

{push(e, j, b1), John(j), button(b1), button(b2), **red(b1)**, current(n1), next(n1, n2), next(n2, n3), ref(root, e), ref(n1, j), **ref(n2, b1)**, distractor(n2, b1), **canadjoin(N, n2)**}

pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: **canadjoin(N, n2), ref(n2, b1), red(b1)**
Effect: ∀y.¬red(y) → ¬distractor(n2, y)

Apply Effects

State:

{push(e, j, b1), John(j), button(b1), button(b2), red(b1), current(n1), next(n1, n2), next(n2, n3), ref(root, e), ref(n1, j), ref(n2, b1), ~~distractor(n2, b1)~~, canadjoin(N, n2)}

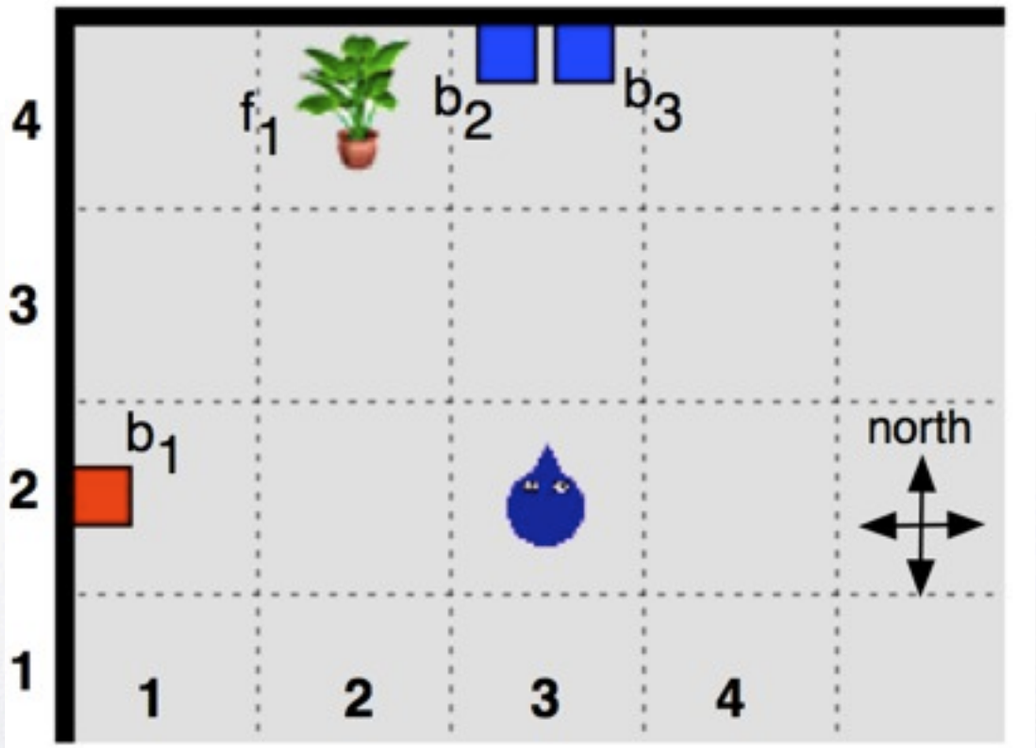
pushes(root, n1, n2, n3, e, j, b1):
Precond: subst(S,root), ref(root,x), push(e, j, b1), current(n1), next(n1, n2), next(n2, n3)
Effect: ¬subst(S,root), subst(NP, n1), subst(NP, n2), ref(n1, j), ref(n2, b1), ∀y.distractor(n1, y), ∀y.distractor(n2, y)

John(n1, j):
Precond: subst(NP, n1), ref(n1, j), John(j)
Effect: ¬subst(NP, n1), ∀y.¬John(y) → ¬distractor(n1, y)

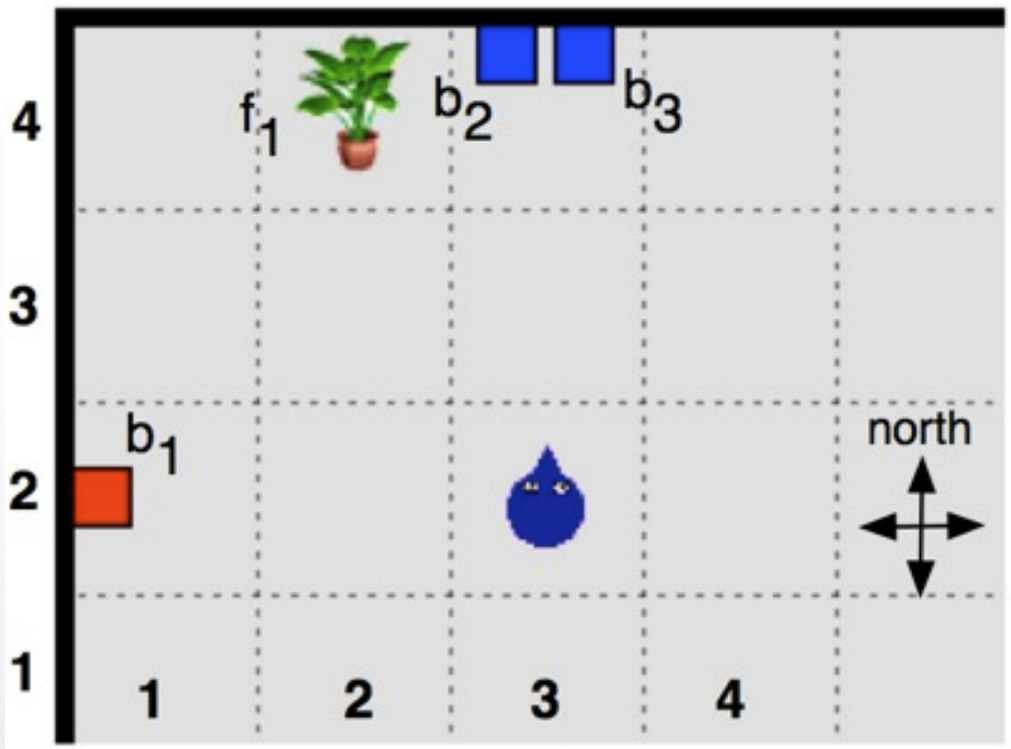
the-button(n2, b1):
Precond: subst(NP, n2), ref(n2, b1), button(b1)
Effect: ¬subst(NP, n2),canadjoin(N, n2), ∀y.¬button(y) → ¬distractor(n2, y)

red(n2, b1):
Precond: canadjoin(N, n2), ref(n2, b1), red(b1)
Effect: **∀y.¬red(y) → ¬distractor(n2, y)**

Simple instruction example



Simple instruction example



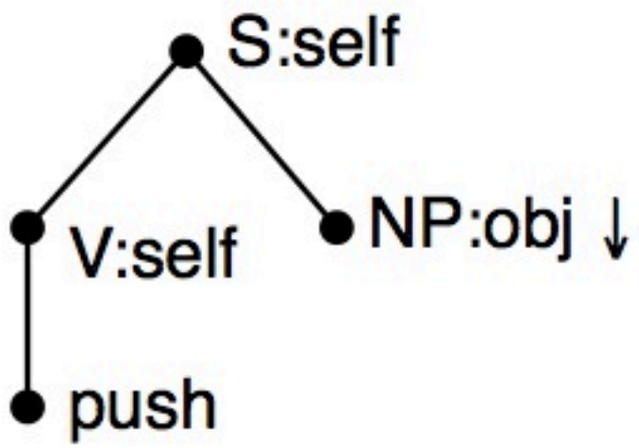
player at pos(3,2)
facing north

goal: push the red button b1
possible instructions:

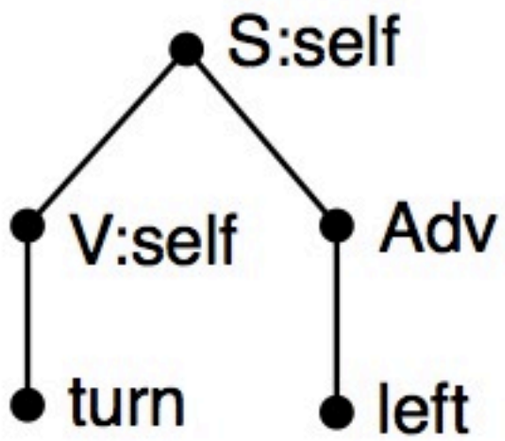
1. "*push the button on the wall to your left*"
2.
   - *"turn left"*
   - *"now push the button in front of you"*

Non-verbal context

- add **preconditions** and **effects** to simulate the physical environment: **visible** objects, **player pos**ition/ **ori**entation



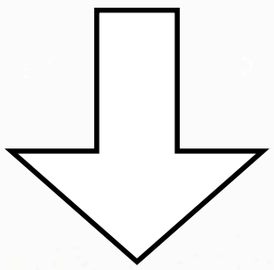
semreq: visible$(p, o,$ obj)
nonlingcon: player–pos(p),
player–ori(o)
impeff: push(obj)



nonlingcon: player–ori(o_1),
next–ori–left(o_1, o_2)
nonlingeff: ¬player–ori(o_1),
player–ori(o_2)
impeff: turnleft

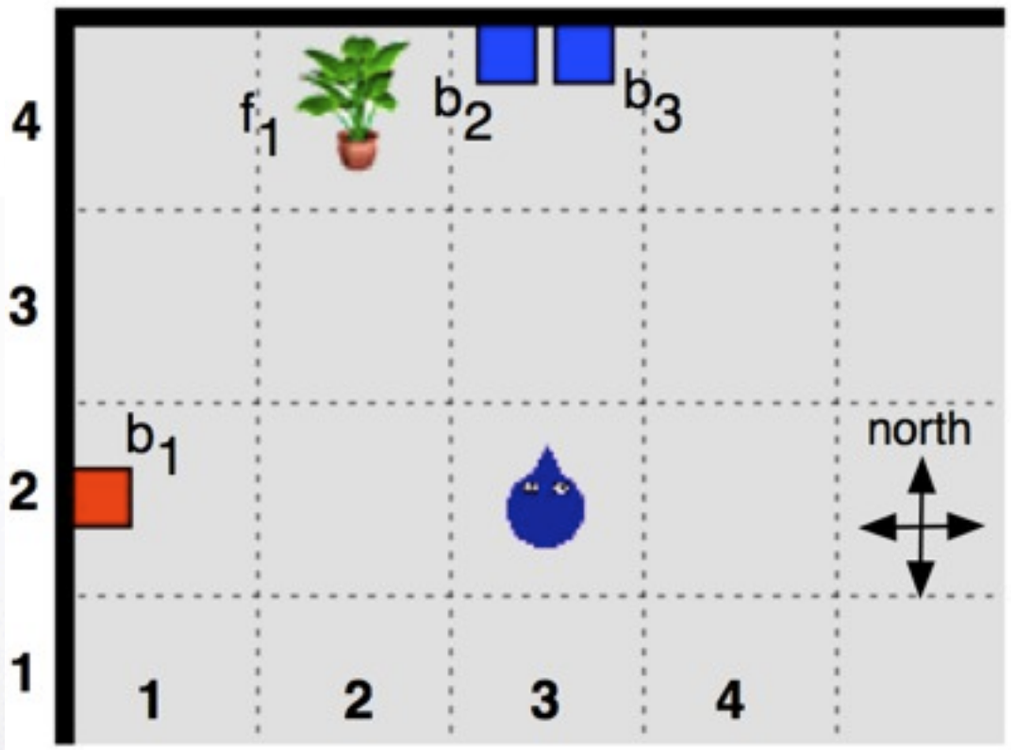
Non-verbal context add-ons

pushes$(u, u_1, u_2, u_n, x, x_1, x_2)$:
Precond: $\mathsf{subst}(\mathrm{S}, u), \mathsf{ref}(u, x), \mathsf{push}(x, x_1, x_2),$
$\mathsf{current}(u_1), \mathsf{next}(u_1, u_2), \mathsf{next}(u_2, u_n)$
Effect: $\neg\mathsf{subst}(\mathrm{S}, u), \mathsf{subst}(\mathrm{NP}, u_1), \mathsf{subst}(\mathrm{NP}, u_2),$
$\mathsf{ref}(u_1, x_1), \mathsf{ref}(u_2, x_2), \forall y.\mathsf{distractor}(u_1, y),$
$\forall y.\mathsf{distractor}(u_2, y)$



push$(u, u_1, u_n, x, x_1, p, o)$:
Precond: $\mathsf{subst}(\mathrm{S}, u), \mathsf{ref}(u, x), \mathsf{player{-}pos}(p),$
$\mathsf{player{-}ori}(o), \mathsf{visible}(p, o, x_1), \ldots$
Effect: $\neg\mathsf{subst}(\mathrm{S}, u), \mathsf{subst}(\mathrm{NP}, u_1), \mathsf{ref}(u_1, x_1),$
$\forall y.(y \neq x_1 \wedge \mathsf{visible}(p, o, y) \rightarrow \mathsf{distractor}(u_1, y)),$
$\mathsf{to{-}do}(\mathsf{push}(x_1)), \mathsf{canadjoin}(\mathrm{S}, u), \ldots$

Context-dependent Adjectives



goal: push b2

instruction:

"*push the left button*"

challenges:

- interpretation of **left** wrt spatio-visual context
- the meaning depends on the modified phrase: **left button** refers the leftmost object among the buttons

Context-dependent Adjectives

left(u, x):

Precond: $\forall y.\neg(\mathsf{distractor}(u, y) \wedge \mathsf{left\text{–}of}(y, x))$,
$\mathsf{canadjoin}(\mathsf{N}, u), \mathsf{ref}(u, x)$

Effect: $\forall y.(\mathsf{left\text{–}of}(x, y) \rightarrow \neg\mathsf{distractor}(u, y))$,
$\mathsf{premod\text{–}index}(u, 2), \ldots$

red(u, x):

Precond: $\mathsf{red}(x), \mathsf{canadjoin}(\mathsf{N}, u), \mathsf{ref}(u, x)$,
$\neg\mathsf{premod\text{–}index}(u, 2)$

Effect: $\forall y.(\neg\mathsf{red}(y) \rightarrow \neg\mathsf{distractor}(u, y))$,
$\mathsf{premod\text{–}index}(u, 1), \ldots$

Adjectives Word Order

How to prevent odd constructions?

large foreign financial firms vs. foreign large financial firms
left red button vs. red left button

Premodifier ordering class-based approach:

- spatial relations: left, upper etc.
- color denoting: red, blue etc.

spatial > color

➡ keep track of the premodifier index

Evaluation

- on the First Challenge on **G**enerating **I**nstructions in **V**irtual **E**nvironments (**GIVE**-1 framework):
  - ➡ perform a **treasure-hunt** task in 3D environment by following real-time instruction
- Test against two baselines
- Word on purpose particularly challenging for REs
  - one room
  - several similar objects and buttons
  - simple description not enough for disambiguation
- Online Evaluation via Amazon Mechanical Turk
  - 25 games for each system

Evaluation

Instructions generated by the three systems in the same scene:

SCRISP	*1. Turn right and move one step.* *2. Push **the right red button**.*
Baseline A	*1. Press **the right red button on the wall to your right**.*
Baseline B	*1. Turn right.* *2. Walk forward 3 steps.* *3. Turn right.* *4. Walk forward 1 step.* *5. Turn left.* *6. Good! Now press **the left button**.*

Evaluation

	success		RE	
	rate	**time**	**success**	**distance**
SCRISP	69%	306	71%	2.49
Baseline A	16%**	230	49%**	1.97*
Baseline B	84%	288	81%*	2.00*

all actions successfully completed

player to referent in steps

Enhancing Referential Success by Tracking Hearer Gaze

(Koller et al. 2012)

Interactive NLG

- **GIVE** Challenge
- more complex virtual environment
- efficient navigation through 3D maze
    - instructor: NLG system
    - player: human
    - ➡ better control
- interactive system giving **feedback** to
    - player's movements
    - player's gaze

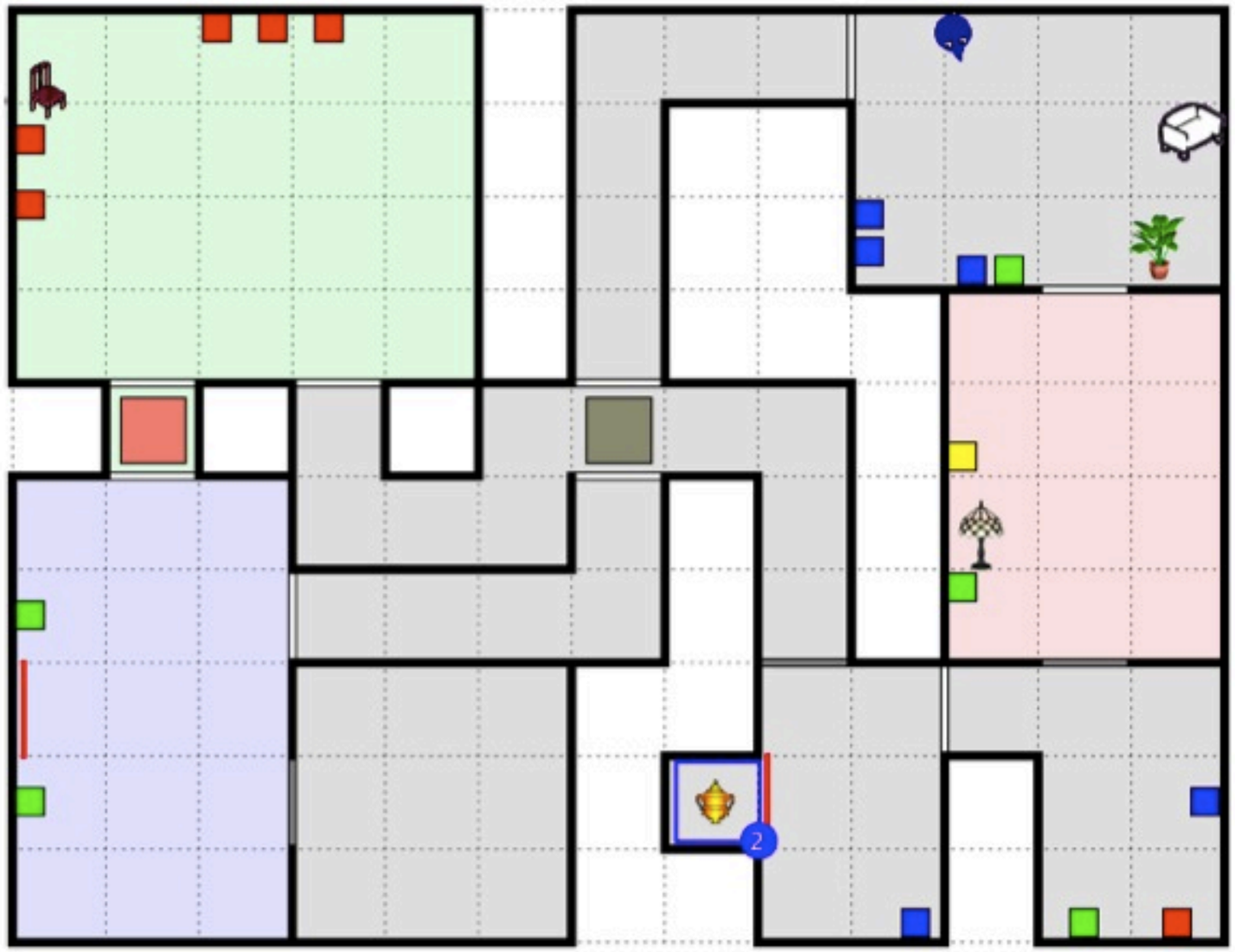
Interactive NLG

- **GIVE** Challenge
- more complex virtual environment
- efficient navigation through 3D maze
    - instructor: NLG system
    - player: human
    - ➡ better control
- interactive system giving **feedback** to
    - player's movements
    - player's gaze

hypothesis:

in situated communication eyetracking facilitates the hearer's reference resolution process

More Complex World



- enter the right room
- press the intended button
- avoid red doorsteps

Monitoring success

- **multimodal** dialogue
  - speaker (the system) gives **verbal** instructions
  - listener (the player) performs **actions** in response

- real time monitoring of player's behaviour
- provided **feedback** to intended action
  - ✓ positive: increases player's confidence
  - − negative: prevents the player to perform wrong action

Three Systems

- **no-feedback:**
  - success not monitored
  - no initiative feedback
  - instruction repetition on request
- **movement-based**
  - monitoring **distance** of the player to a button
  - resolution clue: decreasing distance to a button
- **eyetracking-based**
  - monitoring player's **gaze**
  - resolution clue: fixating an object longer than 300ms

Prototype

Example

System: push the right button to the right of the green button

Player: moves/looks to the intended button

System: **yes that one**

OR

Player: moves/looks to a distractor

System: **no not that one**

relatively vague but facilitates evaluation

Evaluation

Human study:

- 31 participants (12 male)
- all reported English skills: **fluent**
- average age of participants: **27.6**
- experiment duration: approximately **30 mins**
- request help by pressing the **H** key
- questionnaire:
  - noticed eyetracking
  - this information used by the system
  - level of satisfaction wrt interaction

Task and Set Up

- remote monitoring of eye movements on 24-inch monitor
- participants receive written task description in advance
- short practice session
- three complete interactions (Latin square design)

Evaluation

- **Questionnaire:**
  - no particular preference of the tested systems
  - nobody mentioned reaction wrt eye gaze
  - some subjects noticed presence and absence of feedback

- **Confusion:** frequency of pressing the **H** key
  - no-feedback: **2.26**
  - movement-based: **1.77**
  - eyetracking-based: **1.14**
  - ✓ significant difference

- **Referential Success:** ratio of correctly resolved REs
  - first button pressed after hearing a RE
  - player does **not** use the **H** key

- **Scene complexity:**
  - **easy**: no distractors
  - **hard**: all other situations

Results

system	success all	success easy	success hard	success w/out confusion all	success w/out confusion easy	success w/out confusion hard	#scenes all	#scenes easy	#scenes hard
eyetracking	93.4	100.0	90.4	91.9	100.0	88.2	198	62	136
with feedback	94.3	100.0	91.7	92.8	100.0	89.4	194	62	132
without feedback	50.0	-	50.0	50.0	-	50.0	4	0	4
no-feedback	86.6*	100.0°	80.6*	83.5**	98.9°	76.5**	284	88	196
movement	89.8°	100.0°	85.2°	87.5°	97.8°	82.8°	295	92	203
with feedback	93.9	100.0	90.6	91.9	97.7	88.7	247	88	159
without feedback	68.8	100.0	65.9	64.6	100.0	61.4	48	4	44

significant at: ** $p < 0.01$, * $p < 0.05$; ° not significant

communicative success:
onset of first-mention RE to a referent and the participant reaction

Evaluation

- **Further metrics:**
  - number of actions
  - overall route length of the player
  - total duration of interaction
  - time participant spent in idle state

system	#actions (norm.)	distance (norm.)	duration (norm.)	idle (sec)
eyetracking	1.06	1.22	1.49	256.6
no-feedback	1.22*	1.27	1.59	272.5
movement	1.16	1.26	1.56	274.4

Conclusion

Eye gaze:

- reliable clue for hearer referential success
- useful in situated communication
- allows efficient interaction with the user
- provides fast and precise feedback

Summary

- Grammar formalism (TAG) and AI techniques (AP) used for situated **NLG**
- **GIVE** Challenge for Evaluation
- Extension and Improvement Strategy
  - feedback based on hearer eye gaze
  - prototype implementing the strategy
  - user study testing three systems

References

- Garoufi and Koller 2010. "Automated planning for situated natural language generation"
  Proceedings of the 48th ACL, Uppsala.

- Koller et al. 2012. "Enhancing referential success by tracking hearer gaze"
  Proceedings of the 13th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL), Seoul.

Thank you for your attention !

Questions?

Discussion

- Players not aware of eyetracking
  - would the behaviour change if yes?
  - is then gaze still naturally used?

- Other Interesting Applications
- Further Evaluation Metrics