Designing I/O Interfaces for Soar: Possibilities, Guidelines & Interactions

Bob Wray, John Laird and Mike van Lent
robert.wray@umich.edu

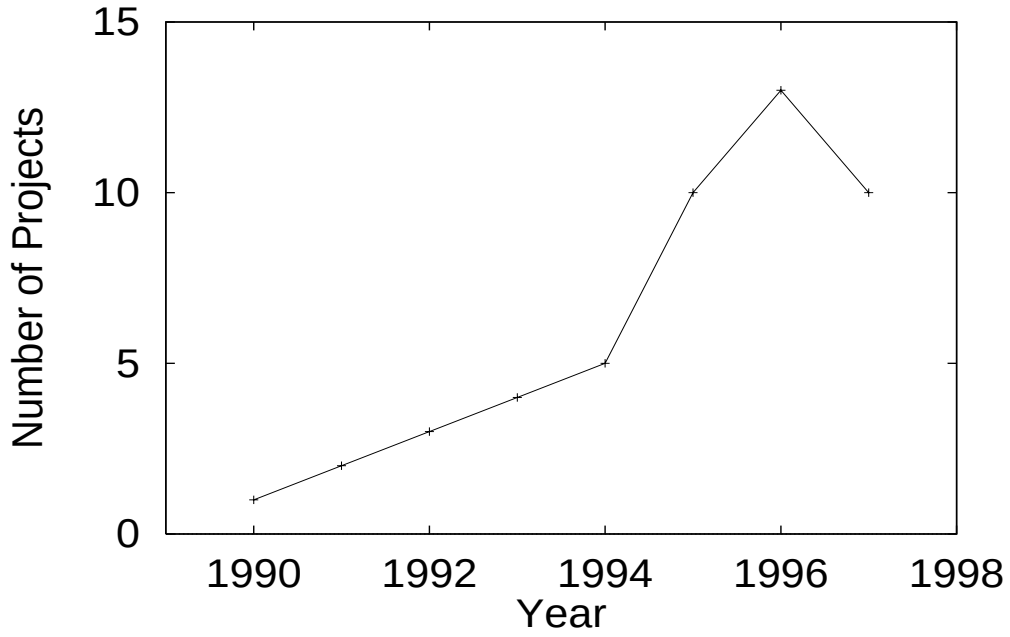
June 1997

New User: “How do I design an external interface for Soar?”

- What are the possibilities?
- What have people done in the past?
  - I/O design more art than science
  - Often an iterative process
- What are the trade-offs between different approaches?

Explore these questions and generate ideas for written guidelines

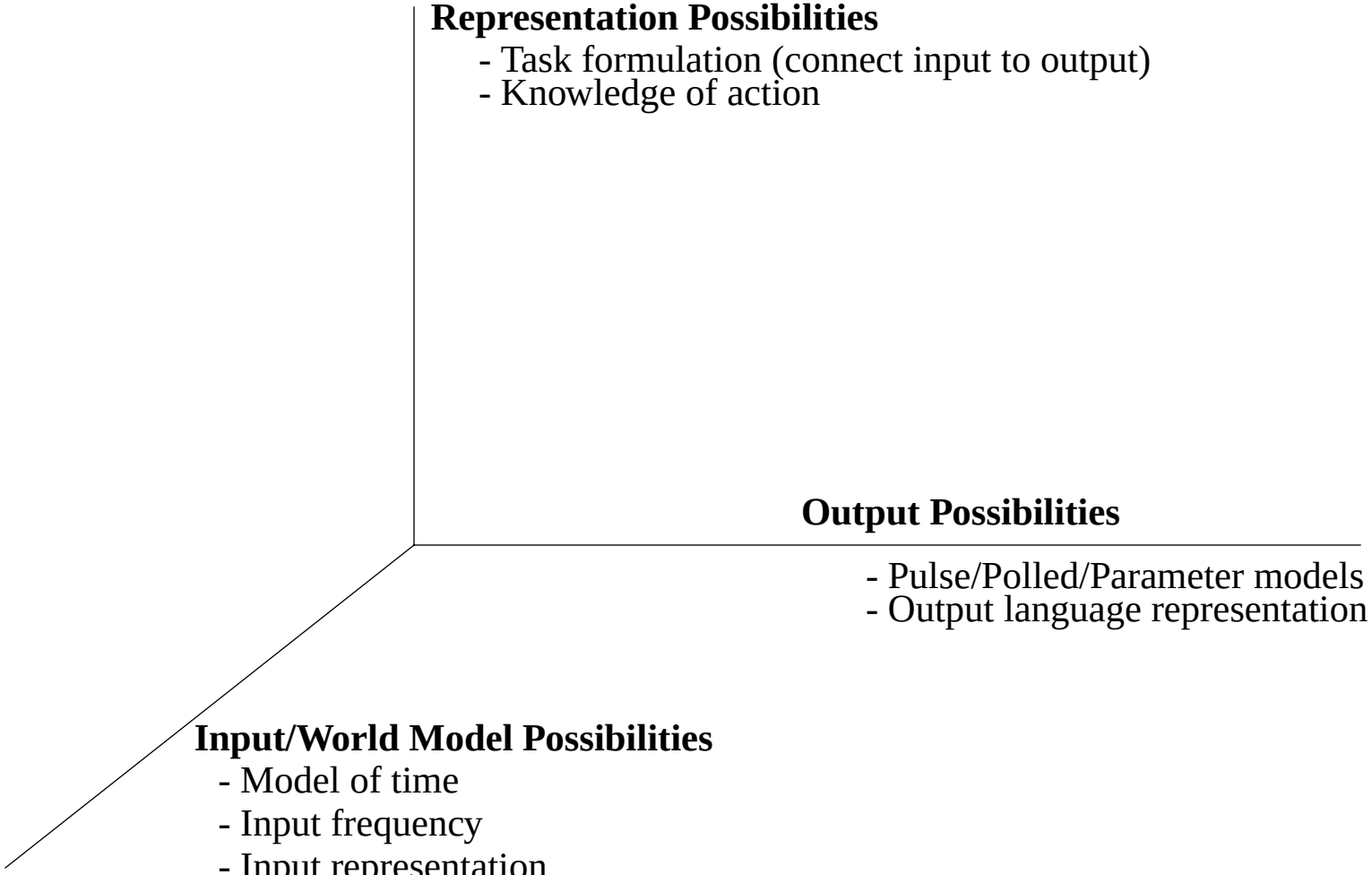
Number of Soar Projects with External I/O



Survey:

“This project used Soar with I/O”	31
Extended Survey	19
Soar Versions: 5, 6, 7, 7+	

Interface Design Space



Explorations in Interface Design

Input:

- Model of time in external environment (19 responses)
  - Isochronous 7
  - Soft real-time (frame-based updates) 1
  - Real-time external environments 6
- Periodicity of input (18 responses)
  - Periodic updates 13
  - Aperiodic/on-demand sensors 6
- Input Representation Language (18 responses)
  - Absolute 11
  - Relative 1
  - Mixed 8

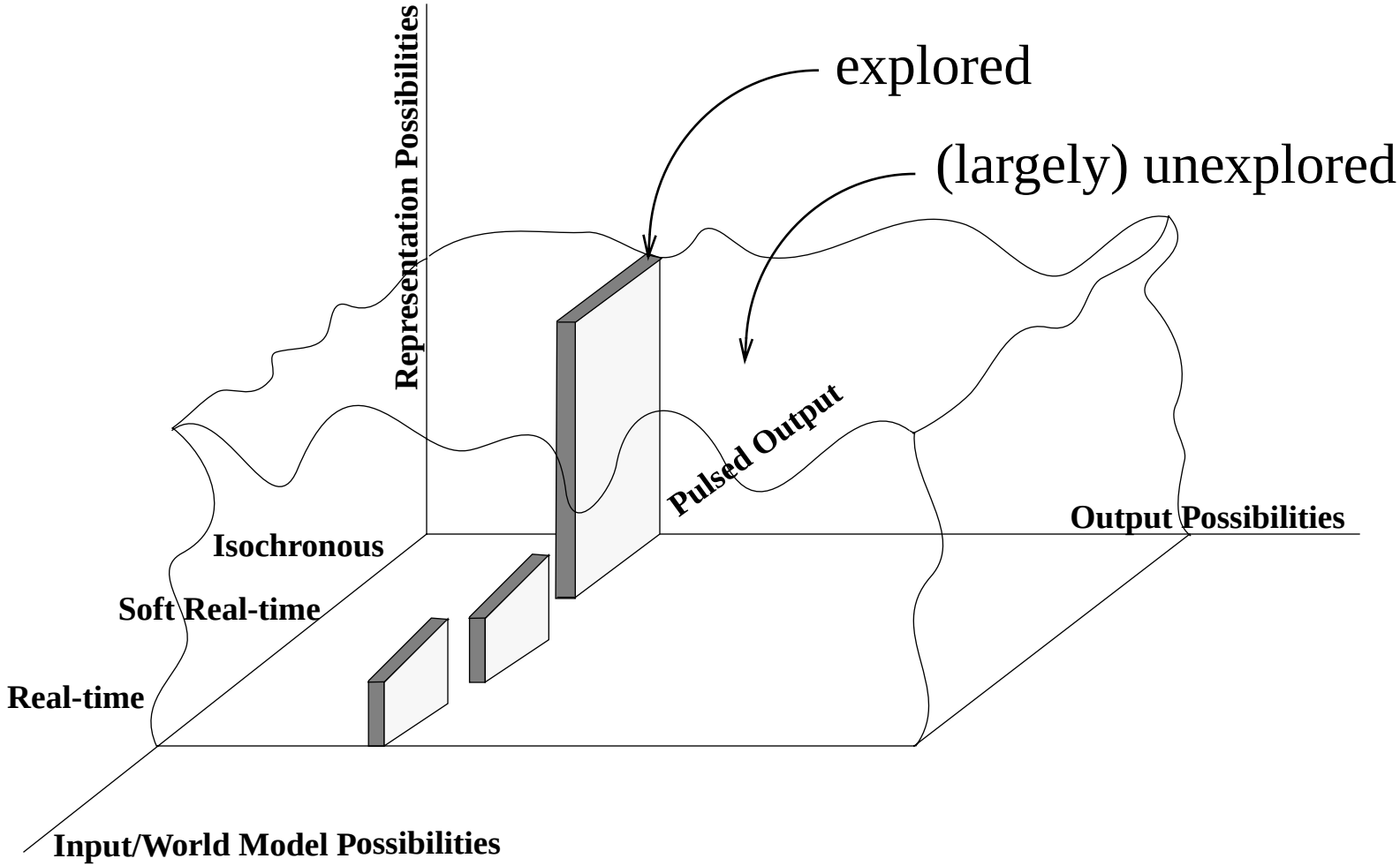
Output:

- Generation of output commands (18 responses)
  - Pulsed Commands 16
  - Other choices 2
- Output Command Language (17 responses)
  - Absolute 6
  - Relative 5
  - Mixed 6

Representation:

- Knowledge of action (19 responses)
  - Unnecessary 7
  - Proprioception 9
  - Production inference 3
  - Mixture of proprioception and inference 1

Explorations in Interface Design(2)



Why these particular choices?

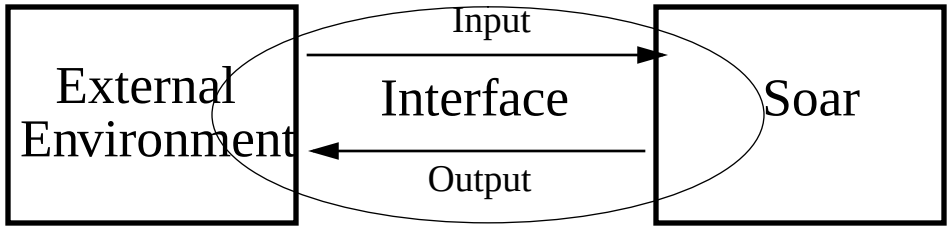
Designer is not free to start from nothing:

1. Constraints imposed by the domain
    - World model
2. Constraints imposed by the architecture
    - Invocation of the output function
3. Constraints imposed by agent goals/behavior constraints
    - Granularity of input
    - Knowledge of action
    - Absolute vs. relative inputs
    - Absolute vs. relative outputs

Constraints Imposed by the Domain

World Model Possibilities

- Isochronal external environment
- Real-time external environment
- Soft real-time external environment
- (Others....)



What are the advantages and disadvantages of each type of model?

Isochronal External Environment

Definition:

External environment changes by a fixed amount each update (regardless of actual processing time)

Examples:

- Many Tcl Interfaces
- Production Models

Advantages:

- Easy (trivial) to synchronize Soar and world

(e.g., need world to progress in 50msec intervals with decisions)

Disadvantages:

- Unrealistic world model for engineering applications
- Real-time processes could be tied to the decision procedure

Real-time External Environment

Definition:

Changes in the external environment occur as soon as the event occurs

Example:

- Air-Soar

Real-time simulation (separate process)

Advantages:

- Real-world applications

Disadvantages:

- Difficult to synchronize with Soar (if necessary)
- World doesn't "know" about Soar's internal state

e.g., decision cycle counter

Soft Real-time External Environment

Definition:

Changes in the external environment are interpolated from a real-time clock (there may be delays)

Example:

- TacAir-Soar

Variable frames based on elapsed time

Advantages:

- Real-world applications
- Possibly easier (than real-time) to synchronize with Soar

Disadvantages:

- Not real-time

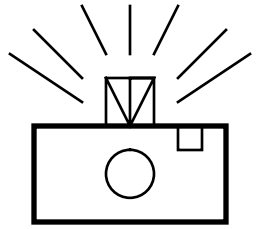
(updates must be within acceptable limits)

Constraints Imposed by the Architecture

Constraint:

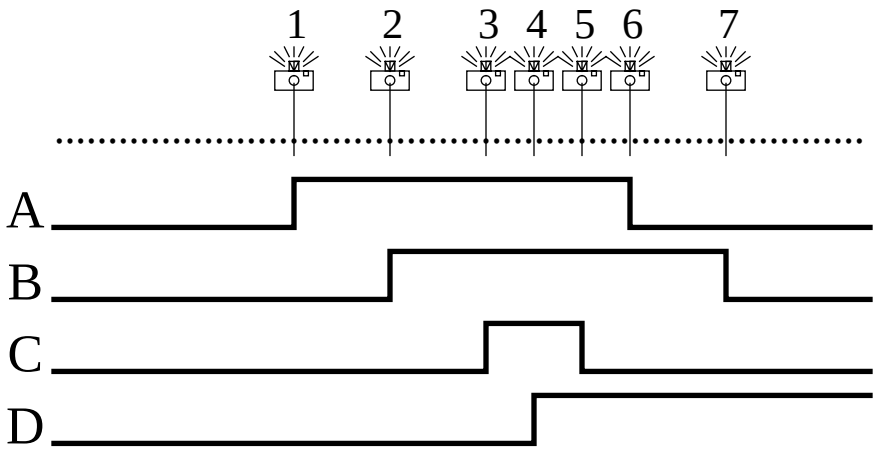
- The output function is invoked only when there is a change to working memory on the output link.
  - OUTPUT PHASE in which a WME has changed

    Output function called; passed entire contents of output-link (camera flash)



  - OUTPUT PHASE in which no output-link WME changed

    Output function not called (no trigger/light provided)

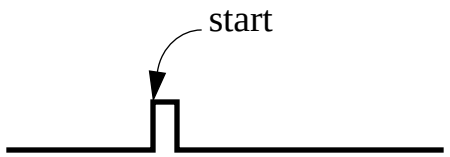


Passed to Output function

.: no call (no WME change)
1: A
2: A, B
3: A, B, C
4: A, B, C, D
5: A, B, D
6: B, D
7: D

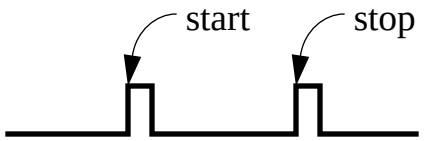
Output-Link Constraint: Options

“Pulse” Options



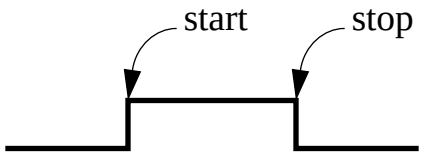
A. Leading edge initiation; self-termination

- Very simple; Soar just initiates command by placing WME on output-link
- Interface/output function must be able to complete command autonomously
- VCR: Push rewind at end of tape



C. Leading edge initiation; leading edge termination

- Low-level control is possible
- Short pulses minimize opportunities for duplication
- Requires knowledge of action
- VCR: Hold fast forward to scan tape



B. Leading edge initiation; falling edge termination

- Better low-level control from within Soar
- Interface must track ‘active’ commands to prevent duplicating commands
- VCR: Push pause to pause tape, press again to resume play

Other options? Possible(?) but difficult due to the output-link constraint

Behavioral Constraints

Goals of the particular agent/model can also impact design choices

- World model constraint is usually obvious with choice of domain
- Choice of Soar → Architecture constraint
- Behavioral constraints require more analysis

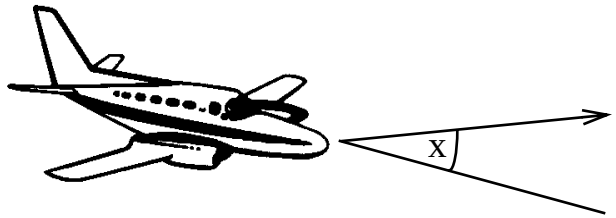
Examples:

- Granularity of input
- Knowledge of action
- Absolute vs. relative inputs
- Absolute vs. relative outputs

Alternatives: Advantages and disadvantages

Granularity of Input

Should a derivable feature be computed by the input system or elaborated from other input values?



(T1 ^side left)

```
if (input-bearing < 0)
→
(item ^side left)
```

1. Elaborate from input with i-supported productions

   Adv: Build intermediate structure

   - important for chunking and data abstraction

   I-support: automatic update of changing features

   DisAdv: Lots of production firings for changing LHS conditions

Granularity of Input(2)

```
[I1 ^thing T1]
[T1 ^input-bearing -10]
[T1 ^side left]
```

2. Input function computes intermediate feature

   Adv: More efficient computation

   DisAdv: Computation done all the time

   (no knowledge of when computation is necessary)

3. On-demand input model

   Adv: Efficient computation done when necessary

   DisAdv: Additional knowledge (sensor knowledge)

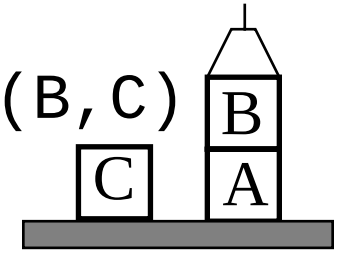
   Additional complexity of output system

No clear best choice: What are the constraints on the agent?

Knowledge of Action

Knowledge of action is necessary when actions have duration.

`stack(B,C)`



"Am I moving?"

```
sp {I*am*moving
 (state <s> ^io.input-link.position <p>
            ^last-recorded-position <> <p>)
 -->
 (<s> ^i-am-moving *yes*)}
```

1. Production-based Inference

   Productions infer status of action by monitoring input

   Adv: Comparatively little constraint on I/O design

   DisAdv: Potentially expensive (lots of productions firing)
   Action effects must be observable

Knowledge of Action(2)

2. Proprioceptive Feedback

Output system provides explicit feedback of action during execution

```
(I6 ^command C4)
(C4 ^type move-block)
(C4 ^status executing)
```

Adv: Low Soar cost (WME update via input)

DisAdv: Strong constraints on I/O interface design

Potential problems due to delays in feedback

Which is better?

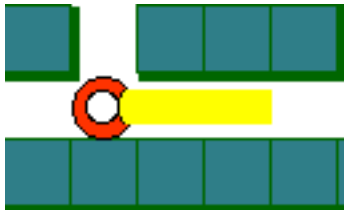
- 3/4's of Soar interface designers use proprioception
- Does that mean anything?

Input Representation

Should the input representation be relative or absolute?

relative

```
(wall ^east 1 ^north 1)
(open ^east 0 ^north 1)
```



absolute

```
(my-position ^x 10 ^y 5)
(wall ^x 11 ^y 4)
(open ^x 10 ^y 4)
```

Relative: Input values are relative to the agent

- Potentially less new WMES (more efficient)
- Poor choice for communication
- Sometimes difficult to detect changes for operator application

Absolute: Input values are relative to some fixed absolute value

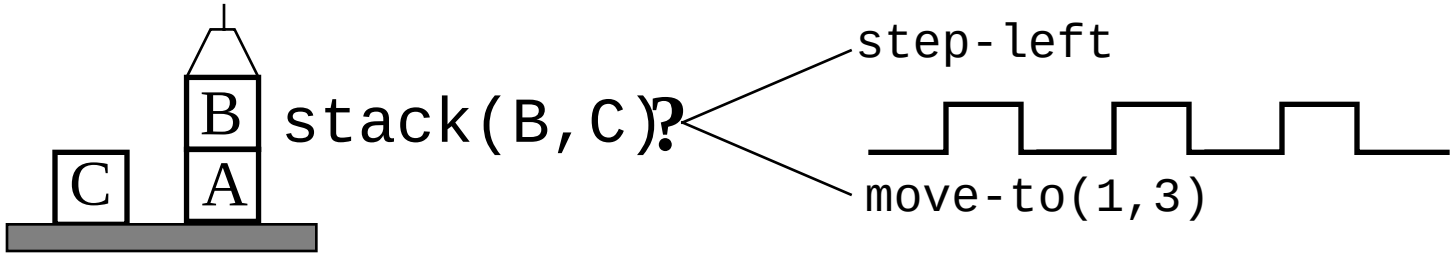
- May require translation to relative values for some reasoning
- Some environments may not have convenient absolute dimensions
- Leads to over-specific chunks

Which to choose? Are agent's goals expressed in relative or absolute values?

May determine how much computation is necessary from input

Output Command Language

Should output commands refer to absolute or relative values in the world?



Relative:

- Knowledge of action critical for relative values when interface does not store currently executing commands (duplication problem)

Absolute:

- Identical, multiple commands not generally a problem
- Over-specific chunks (without translation at the top level)

Which to choose?

What is agent's input representation language?

Conclusions

Increased Awareness of I/O Interface Design Issues

- Improve understanding of obvious design constraints

  Domain constraints

  Architecture constraints

- See other design choices in terms of agent/behavior constraints
- Better able to help new users design their systems

  What are interactions to be aware of when designing an I/O interface?

  I/O Design “Rules of Thumb”:

  suggestions/comments/ideas: robert.wray@umich.edu

Should/could architecture be altered to accommodate larger parts of design space?

By understanding the design dimensions, costs of our solutions, and potential costs of other alternative solutions, we can better tailor the architecture to the demands of interfacing with external worlds.