

The context problem for software agents



Software agents provide assistance to users of interactive graphical interfaces

What should I do for the user?

How should I do it?

Agent needs to understand the intention of the user

Intention often can be inferred from context

Context reduces explicit input

Agent Personalities



Butler / Servant / Secretary

- Agent executes commands, satisfies goals, anticipates needs

Teacher / Student, Master / Apprentice

- Agent learns skills taught by user

The context problem for learning agents



Agent can only learn from concrete experience

Concrete experience needs to be *generalized*

Conservative: Stick close to experience

Increased Accuracy

Liberal: Try to do as much abstraction as possible

Increased Applicability

Programming by Example [or "by Demonstration"]



Agent "watches what you do" in the interactive interface

Records sequence of operations, data involved in operation

Generalizes program so that you can use an analogous procedure in new examples

Strategies for generalization in Programming by Example



System makes a guess heuristically

Inferred from context, domain-dependent

System asks user

System may supply choices to give user context

System receives advice from user

Advice used as context for system's choices

The Data Description Problem



[Halber]

How should objects involved in examples be described?

Intentional vs. extensional descriptions

Hierarchical descriptions

Machine Learning: Version Spaces

Also: Action Description Problem, generally easier

The Critique Problem



Don't do *that* again!

What's *that*?

Examples of PBE systems



Mondrian Graphical Editing

Grammex Text Recognition Agents

Agent Personalities



Butler / Servant / Secretary

- Agent executes commands, satisfies goals, anticipates needs

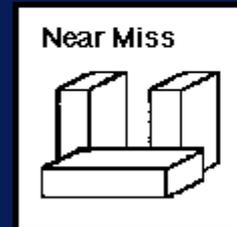
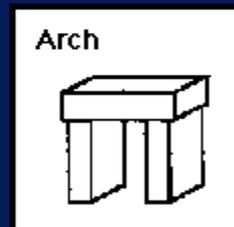
Teacher / Student, Master / Apprentice

- Agent learns skills taught by user

Mondrian: An Instructible Graphical Editor

Mondrian=

- An object-oriented graphical editor +
- An agent that records user interface actions +
- Learning and generalization heuristics



Mondrian: An Instructible Graphical Editor

The image displays the Mondrian graphical editor interface, which is used for creating and editing graphical user interfaces. The interface is divided into several panels:

- Toolbars:** Located on the left side, these include:
 - New Command:** A toolbar with icons for creating new commands.
 - Rectangle:** A toolbar with icons for creating and editing rectangles.
 - Remove Ring:** A toolbar with icons for removing rings from objects.
 - Select:** A toolbar with icons for selecting objects.
 - Move:** A toolbar with icons for moving objects.
 - Delete:** A toolbar with icons for deleting objects.
 - Copy:** A toolbar with icons for copying objects.
 - Group:** A toolbar with icons for grouping objects.
 - Resize:** A toolbar with icons for resizing objects.
 - Color:** A toolbar with icons for changing the color of objects.
 - Line:** A toolbar with icons for creating and editing lines.
 - Text:** A toolbar with icons for creating and editing text.
 - Annotate:** A toolbar with icons for adding annotations to objects.
 - New Part:** A toolbar with icons for creating new parts.
 - Assemble:** A toolbar with icons for assembling objects.
 - Disassemble:** A toolbar with icons for disassembling objects.
- Main Canvas:** The central area where the graphical user interface is being edited. It shows a large image of a car engine with a blue background. A label "(AN 'ENGINE)" is positioned below the engine image. A smaller image of the engine body is shown to the right, with a label "(An 'Engine-Body)".
- Storyboard for "Remove Ring":** A panel on the right side of the interface that displays a sequence of frames for the "Remove Ring" command. The first frame shows the engine image with a small red box around it. The second frame shows the engine image with a larger red box around it. Below the frames, the text "Disassemble" is displayed, followed by the instruction: "Disassemble the ring of the first argument from the engine body of the first argument."

Web has renewed interest in text parsing technology

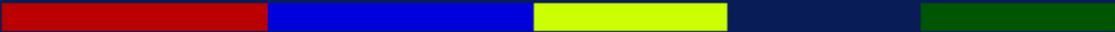


Much information on the Web and in desktop applications exists in “semi-structured” form

- Structured data embedded in unstructured data

Higa --
Phone Number
Great news -- a ticket to Moorea came through! Please call our travel agent at 415-555-9662 and give him your passport number. I'm going to be on the road this afternoon, but you can leave a message with my secretary at [redacted]@apple.com. **E-Mail Address**
Also -- check out the lagoon views at [http://www.\[redacted\].com/islands/moorea/main.html](http://www.[redacted].com/islands/moorea/main.html) -- really gorgeous looking scenery. Can't wait to get there!
Later,
John

Parsers are controlled by grammars



Grammar is a set of rules, each of which recognizes a class of text strings

Usually written in BNF or equivalent

Users have difficulty writing in a formal language

... but they DO understand the concepts behind recognition

“An e-mail address is [usually] a person’s name, followed by an “@”, followed by a host”

Solution: Define grammars by example!



Grammars are difficult because they are abstract

It's hard to understand what the effect of writing a rule will be in particular examples

Stresses short-term memory, reasoning

People are much better at dealing with concrete examples than abstractions

So, present concrete textual examples ~~and~~ the system how to interpret them

Grammex = “Grammars by Example”



A “programming by example” interface for defining grammars

User supplies example of text to be recognized

System tries to parse text according to current grammar

User can interactively specialize and generalize interpretations of substrings

Grammex compiles a BNF-like grammar

Grammex



Grammex rule windows

