Minstrel: A case-based story generator

- Minstrel is an author modeling story generator
  - Pursues storytelling goals, looking up solutions in its case memory
- Minstrel is a model of creativity
  - High level assumptions about creativity
    - Creativity is driven by a failure of problem solving
    - Creativity is an extension of problem solving
    - New solutions are created by using old knowledge in new ways
- Minstrel used in two different domains
  - Storytelling – the primary domain, generates King Arthur stories
  - Mechanical design

Case-based reasoning

- Many of you may be familiar with a 0th order version of case based reasoning – k nearest neighbor
  - Recall is simple distance metric, often Euclidean
  - No adaptation, simply lookup (majority and averaging are simple adaptation)
  - No assessment
  - Minstrel is an instance of symbolic CBR – much richer than kNN
  - CBR systems learn by storing new cases
Problem solving failure in CBR

- CBR problem solving failure can occur because:
  - Fail to find a solution to a similar problem
  - Fail to adapt solution
  - Fail domain assessment

- Additionally, as a model of creativity, Minstrel fails if it is bored
  - Boredom occurs if a proposed solution has been seen too many times (used at least twice)

- Cases stored in episodic memory – a memory of events and experiences indexed by salient cues (e.g. location, action performed, etc.)
  - Forms classes of related episodes

Challenges of creativity

- Need to modify CBR to account for creative problem solving

- New solutions require knowledge not indexed under current problem
  - Creativity begins when problem solving fails – in CBR, this means that no solutions are found for current problem, or adaptation runs into a dead end

- Creativity involves recasting the problem
  - If trying to solve the current problem is failing, try solving a related problem (search problem space, not just the solution space)

- Adaptation must avoid too much complexity
  - Solving arbitrarily different problems, then trying to adapt that to the original problem, won’t work (limits on creativity)

Example problem space

![Figure 2.2 Search Space of Slightly Different Problems](image)
TRAMs – reifying problem space search

- TRAM – Transform Recall Adapt Method.
  - A bundled knowledge representation that knows how to:
    - Transform a problem description into a related description
    - Recall a solution to the new problem
    - Adapt the solution back to the original problem

- TRAM: Cross-Domain-Solution
  - Transform strategy: find a new problem domain with similar actions and actors, and analogically map current problem to new domain
  - Adapt strategy: Map any discovered solutions back to the original domain by reversing analogical mapping

Example of TRAM: Cross-Domain-Solution

- Domains the system knows about
  - Knights and princesses (King Arthur)
  - Businessmen and friends

- Original Problem: create a scene in which a knight accidentally meets a princess
- Transformed Problem: create a scene in which a businessman accidentally meets somebody
- Recalled scene: "John was sitting at home one evening when his dog began acting strange. The dog was scratching at the door and whining for a walk. Finally, John decided to take the dog for a walk. While they were out, John ran across his old friend Pete, whom he hadn’t seen in many years."
- Adapted solution: "One day while out riding, Lancelot’s horse went into the woods. Lancelot could not control his horse. The horse took him deeper into the woods. The horse stopped. Lancelot saw Andrea, a Lady of the Court, who was picking berries."

Problem solving with TRAMs

[Diagram of problem solving with TRAMs]
Imaginative memory: recursive TRAM application

- Leaps in creativity result from combinations of small modifications

Inventing suicide: example TRAM application

- Episodic memory contains two episodes

Knight Fight
A knight fights a troll with his sword, killing the troll and being injured in the process.

The Princess and the Potion
A lady of the court drank a potion to make herself ill.

Storytelling goal to solve

Application of TRAM: Generalize-Constraint

- TRAM: Generalize-Constraint
  - Transform: select and generalize a feature (call it $generalized$-feature) of the scene specification. Use this new scene specification as an index for imaginative recall.
  - Adapt: adapt the recalled solution to the current problem by adding $generalized$ feature back to the recalled scene
**Generalizing features using class hierarchy**

- TRAM:Generalize-Constraint tries to use most specific generalization that recalls a solution.
- If we completely removed the constraint, can have creativity failures.
  - Example: solve "A knight gives a princess something to make her happy."
  - Transform: "A knight gives somebody something to make them happy."
  - Recall: "A knight gives a troll a hunk of meat to make it happy."
  - Adapt: "A knight gives a princess a hunk of meat to make her happy."

**Example using recursive TRAMs**

- TRAM:Similar-Outcomes-Partial-Change
  - Transform: Replace the resulting state caused by an action with a related state.
  - Adapt: Replace state change in recalled episode with original state.
- Apply TRAM:SOPC to "A knight purposefully kills himself" to create "A knight purposefully injures himself".
  - Recursively apply TRAM:GC to "A knight purposefully injures himself" to create "Someone purposefully injures themselves."
  - Recall: "A lady of the court made herself ill by drinking poison."
  - Adapt (using TRAM:GC) to "A knight made himself ill by drinking poison."
  - Adapt (using TRAM:SOPC) to "A knight killed himself by drinking poison."

**Randomness**

- In AI models of creativity, we want to minimize the effect of randomness.
  - Random is not the same as creative.
- In Minstrel, random decisions are made only when
  - Multiple episodes are recalled from a single query.
  - Multiple author plans or TRAMs are applicable.
- These random decisions are highly constrained by the problem solving process.
  - There’s a lot of process on top of the randomness.
- Compare with the use of randomness in
  - Context free grammars
  - Mad Libs.
Minstrel is driven by author-level goals

- The preceding slides described the solving of character-level goals
  - The character goals were set by an author to serve the story, but aren’t themselves directly goals about the form and content of the overall story
- Minstrel’s author goals
  - Thematic goals – the stories illustrate a theme, in Minstrel’s case, Planning Advice Themes (e.g. “A bird in the hand is worth two in the bush.”)
  - Drama goals – goals regarding the unity of action (tragedy, foreshadowing)
  - Consistency goals – motivate and explain story actions
  - Presentation goals – goals about which events must be fully described, and which can be summarized or omitted (in general diegetic goals)

Author-level processes

- Author planning is broken into two processes
  - Goal management
  - Problem solving
- Author-level problem solving is accomplished exactly the same way as character-level problem solving

Representations of goals and plans

- Character and author-level goals use the same representation
- Character and author-level plans are represented differently
  - Character-level plans are represented using explicit declarative schema (we’ve seen examples earlier in the lecture)
  - Author-level plans are represented using Lisp
    - Avoids having to invent yet another representation of computation
    - But makes author-level plans a black box, hugely limiting creative problem solving
Example author plan

- **Name**: ALP:Make-Consistent-P-Health
- **Goal Type**: &Check-Consistency
- **Object**: "AL-OBJ"
- **Test**: "AL-OBJ" is a state schema that represents a character being wounded, and it is not motivating a goal by that character to protect his health
- **Body**:
  1. Create &P-Health (health protection) goal for the character being wounded in "AL-OBJ"
  2. Add a new goal to the story by creating a &Thwarts link from "AL-OBJ" to the new goal
  3. Create author-level goals to make sure the new goal is consistent and fully instantiated

Nonadaptive creativity

- Since author-level plans can’t be directly adapted, the system author (meta-author) provides generalized default versions of author plans that are guaranteed to work
  - Example: the generalized ALP:Default-Consistent plan says that, for whatever object it’s ensuring consistency for, the object (e.g. state, goal) is consistent if it has a type
- There’s a special TRAM that recalls default author plans if the standard recall TRAM doesn’t work
  - See page 85 for a snippet of the index tree for author plans

Minstrel conclusion

- Minstrel provides a case-based model of creativity, with specific support for storytelling via the author-level goals and plans
- Imaginative memory (recursive TRAM application) is used to instantiate story episodes that meet requirements set by author-level plans
- Author-level problem solving uses the same TRAM mechanism as character-level problem solving, though creative transformation is greatly limited by the black box nature of author plans
- Authoring within the Minstrel architecture means specifying
  - An ontology for character actions (what are the types of states, objects and character goals in the world)
  - Initial character episodes (stories) known to the system
  - TRAMS (these are relatively story domain independent)
  - Author goals and plans (these are relatively story domain independent)