Tale-Spin: world and character simulation

- Tale-Spin was the first world and character simulation approach to story generation
- A story is generated as a consequence of character pursuing plans to accomplish goals
- The world simulator automatically infers consequences of actions taken by characters

Example story

- Let’s look at an example story generated by micro-talespin, the story of thirsty Irving and stubborn Joe ("story2")
- Let’s look at the output for where we see character and world modeling happening
Knowledge used by the simulator

- Goals and plans: there is a collection of plans to be used by the characters in accomplishing goals
  - There are alternate plans for accomplishing the same goal.
  - Plans have preconditions for when they are appropriate.
  - Plans can initiate subgoals.
- Actions: there is a primitive set of actions known by the simulation
  - Used conceptual dependency (CD), an ontology of actions used by the NLP and narrative research of Roger Schank's research group
- Characters: characters possess goals (which they look up plans to accomplish) and have a memory of facts they know
- Inferences: a collection of rules for inferring the consequences of knowledge and actions
- Natural language generation rules

Conceptual dependency (examples)

- Atrans – transfer of possessions (object) from one agent to another
- Grasp – an agent picks up an object or drops an object
- Ingest – an agent eats an object
- Mbuild – build new knowledge out of old
- Mtrans – transfer knowledge from one agent to another
- Propel – to apply physical force to an object
- Ptrans – to transfer the physical location of an object

Example plans for getting an object

- DCont (GetObject) – to get an object, if you know someone has it, persuade them to give it to you, otherwise try to find where the object is, go there and take it
  - DCont succeeds trivially if the agent already has the object
- DCont-1 (GetObject through persuasion)
  - Persuade actor owner (atrans owner object actor) actor should persuade owner to have owner transfer object to actor
- DCont-2 (GetObject by going where it is and getting it)
  - DKnow actor (where-is object) actor knows where object is
  - DProx actor actor object the actor should move themselves near object
  - DAtans actor object actor actor should atrans object to themselves
Example plan for persuasion

- Persuade – you can persuade someone to do something by either asking them, giving them food, or threatening them.

- Bargain-plan – a plan for agent1 to bargain with an agent2
  - **Precondition**: we can only use this plan if agent1 know that it is not the case the agent2 is deceitful towards the agent1, if the agent1 knows the agent2 doesn’t have food, and if agent1 doesn’t have the goal of having food.
  - Build agent1 (cause atrans-food (maybe action)) agent1 stores a fact in its head that it’s hoping that giving food will result in the desired action
  - Tell agent1 agent2 (question (cause atrans-food (future action))) – agent1 asks agent2 if giving agent2 food will result in agent2 performing the desired action
  - Dont agent1 food agent1 has goal to get the food
  - Dpros agent1 agent1 agent2 agent1 has goal to move itself near agent2
  - Atrans agent1 food agent2 agent1 gives food to agent2
  - Test if action is true – did agent2 keep the bargain?

Consequences

- Actions have consequence rules associated with them.

- Example: Atrans-conseq
  - Everyone in the area notices that the agent performed the atrans
  - Everyone in the area notices that the receiving agent possesses the object
  - Everyone in the area notices that the object is physically held by the receiving agent
  - Everyone in the area notices that the giving agent no longer possesses the object

- The Atrans consequences are an example of primitive world and agent “physics” – bookkeeping performed to know who know what.

Consequences model social action

- Besides bookkeeping, consequences are used to model social action.

- Promise-conseq – consequences of y asking x to do xdo after y performs ydo
  - If x is deceitful towards y, then x will tell y they are stupid after y performs the action (sets up a demon) and tells y this
  - Otherwise x says no (they won’t perform xdo)

- Builds the rules of social action into the world
  - Difficult to have agent-specific responses
Model of storytelling

- For Tale-Spin, a story is the result of agents pursuing plans in the face of goals.
- Let’s compare this with Ryan’s 8 narrative dimensions:
  - Spatial and temporal dimension met easily (individualized existents, significant transformation, non-habitual action)
  - Mental dimensions are met (some of the participants are intelligent agents who pursue planful activity motivated by goals)
  - Pragmatic dimensions are a problem:
    - No unified causal chain leading to closure – must carefully set initial conditions to establish this
    - The story actions are asserted as facts
    - System doesn’t explicitly reason about meaning of story
- How would we “interactivize” Tale-spin?

Hierarchical planning and characters

- Cavazza et. al. is a modern incarnation of a character and world-modeling approach to story generation.
- They employ character-centric hierarchical task planning to a Friends domain.
- First we need to have some idea of what hierarchical task planning is.

Motivation for HTN Planning

- We may already have an idea how to go about solving problems in a planning domain.
- Example: travel to a destination that’s far away:
  - Domain-independent planner:
    - many combinations of vehicles and routes
  - Experienced human: small number of “recipes”
    - e.g., flying:
      1. buy ticket from local airport to remote airport
      2. travel to local airport
      3. fly to remote airport
      4. travel to final destination
- How to enable planning systems to make use of such recipes?
HTN Planning

- Problem reduction
  - Tasks (activities) rather than goals
  - Methods to decompose tasks into subtasks
  - Enforce constraints
    - E.g., taxi not good for long distances
  - Backtrack if necessary

Simple Task Network (STN) Planning

- A special case of HTN planning
- States and operators
  - The same as in classical planning
- Task: an expression of the form \( t(u_1, \ldots, u_n) \)
  - \( t \) is a task symbol, and each \( u_i \) is a term
- Two kinds of task symbols (and tasks):
  - Primitive: tasks that we know how to execute directly
  - Task symbol is an operator name
  - Nonprimitive: tasks that must be decomposed into subtasks
    - use methods (next slide)
Methods

- Totally ordered method: a 4-tuple
  \( m = (\text{name}(m), \text{task}(m), \text{precond}(m), \text{subtasks}(m)) \)
  - \( \text{name}(m) \): an expression of the form \( n(x_1, \ldots, x_n) \)
  - \( x_1, \ldots, x_n \): parameters - variable symbols
  - \( \text{task}(m) \): a nonprimitive task
  - \( \text{precond}(m) \): preconditions
  - \( \text{subtasks}(m) \): a sequence of tasks \( \langle t_1, \ldots, t_k \rangle \)

\[ \text{air-travel}(x,y) \]
\[ \text{task: travel}(x,y) \]
\[ \text{precond: long-distance}(x,y) \]
\[ \text{subtasks: } \langle \text{buy-ticket}(a(x), a(y)), \text{travel}(x, a(x)), \text{fly}(a(x), a(y)), \text{travel}(a(y), y) \rangle \]

Limitation of Ordered-Task Planning

- Cannot interleave subtasks of different tasks
- Sometimes this can make things awkward
- Need methods that reason globally instead of locally

Generalize the Methods

- Generalize methods to allow the subtasks to be partially ordered
- Consequence: plans may interleave subtasks of different tasks
- This makes the planning algorithm more complicated
Applying HTN planning

- Domain: methods, operators
- Problem: methods, operators, initial state, task list
- Solution: any executable plan that can be generated by recursively applying
  - methods to nonprimitive tasks
  - operators to primitive tasks

Application of HTN to storytelling

- They interleave planning and execution
  - Backtrack when a primitive action fails
  - This is different than traditional HTN planning that does a full forward search
- Use total ordering of task nets
  - Argue that, since stories are structurally decomposable into unique pieces, don’t need intermixing
  - But their formalism combines character and story-level distinctions, resulting in characters who can only do one thing at a time
    - Problem for believability
- HTNs, as a knowledge-rich planning formalism, is appropriate for storytelling
  - Internal nodes (task nets) can implicitly encode “desired world changes” that aren’t explicitly captured in the domain ontology (procedural vs. declarative encoding)