

Notes on Attention and Working Memory

Jim Mason – February-March 2012

What entities are in this situation?

- agents – other and self
- cards, their positions, and their orientations (up or down)
- card piles (collections of overlapping cards)
- locations on card table
- quantities
- utterances by an agent to an agent

What events (changes in the situation) occur?

- moving of cards and piles (by an agent)
- changing the sequence of cards in a pile (e.g., shuffling)
- combining or splitting of piles (as a result of moving cards or piles)
- turning cards or piles over (by an agent)
- stacking or spreading cards in a pile (by an agent)
- pointing to cards (or indirectly to piles) (by an agent)
- pointing to locations on the card table (by an agent)
- new utterances (by an agent to an agent) – interpreted as request, ask, or tell

What events occurred before the current ones? (memory)

Actually, memory will consist of some kind of chain(s) of situation states and events that change(d) states to new states. See Framing the Situation, later.

What entities and events are an agent (self or other) attending to? (focus)

Note: Focus is involved in interpreting deictic and anaphoric references. Focus is usually on the most recent event and the entities involved in that event. In that case, it is *event driven*. Focus can also be *goal driven*, by an agent's internal goals.

Mental ideas (abstractions) of entities

A CardPlayer, as contrasted with a CardGod, attends to events and entities by instantiating *representations* (ideas) of those events and entities, which may be

incomplete. For example, a CardIdea has a Suit, Rank, Orientation, Location, and Pile, some or all of which may be unknown at a given time. Similarly, a CardCollectionIdea is a collection of CardIdea instances, whose exact membership, and sequence may not be completely known. A CardPileIdea is a CardCollectionIdea that also has a(n approximate) Location, Size (quantity of cards), and Radius.

Idea instances may be associated temporarily with perceived entities in the world. Attention focus operates through idea instances, and memory contains idea instances.

Model of a CardPlayer's behavioural cycle

Behaviour of a CardPlayer will be driven by a **goal priority queue** together with a **noticed event queue**. The short-term memory capacity of a CardPlayer will be determined by the maximum sizes of those queues.

In the absence of parallel processing, the dialogue between a CardPlayer and human will be conducted by *turn-taking*. After initialization of data structures, the basic algorithm for a CardPlayer on its turn will be as follows:

if any events have occurred since the previous turn **then**

 create goal(s) and eventIdeas to deal with those events, as necessary;

 put the eventIdeas into the noticed event queue;

 put the goal(s) into the goal priority queue

endif

 Take one (or more) goal from the top of the goal priority queue and act on it, putting a corresponding eventIdea into the noticed event queue, along with a reference to the goal that triggered the event.

Events created by the CardPlayer itself need not necessarily give rise to new goals. The number of goals taken from the top of the goal priority queue on each turn may depend on how full the goal priority queue has become. Normally, enough goals must be dealt with in order to leave capacity in that queue for the number of new goals that are likely to be added before the next turn.

In addition to EventIdeas, which are maintained in the noticedEventQueue, ideas of other entities in the CardWorld may be *permanent* (e.g., ideas of Agents and of PlayingCards without their current locations) or *temporary* (e.g., ideas of CardPiles and possibly incomplete ideas of PlayingCards with current locations) . Memory for temporary ideas may be maintained in a queue of **focused entity sets** supplemental to the noticedEventQueue.

Program objects needed for a CardAgent model of attention and memory

- “real world” entities: CardTable, PlayingCard, Suit, Rank, Color, CardCollection, CardPile, CardTableLocation, CardAgent, CardPlayer, Utterance, etc. Note: Suit and Rank are really not real-world properties, but here refer to the corresponding markings on cards, which are *interpreted* as suit and rank symbols by a CardAgent.
- “real world” events: CardMoveBy, CardTurnBy, PileMoveBy, PileTurnBy, PileShuffleBy, PileSpreadBy, PileStackBy, pointToCardBy, pointToPileBy, pointToLocationBy, utteranceBy, etc.
- “real world” relationships: cardOverlaps, pileOverlaps, eventConcurrentWith (e.g. (PointToCardBy, utteranceBy) or (CardMoveBy, CardTurnBy))
- “unconscious” entities in each CardAgent: abilities, goals, ideas, perceivedEventList, goalPriorityQueue, thingsToBeNoticed priority queue (see below), thingsNoticed list (see below).
- “unconscious” events in each CardAgent: adopted a goal, caused an event, satisfied a goal (e.g., understood an utterance, granted a request), etc.
- “conscious” entities in a CardAgent: an Xidea entity type for each object type X in the preceding categories. Note: Instances of these will be lazily evaluated. They will represent things in the conscious attention and memory of the agent.

Self-awareness in a CardAgent

A CardAgent will have Xidea instances for, among other things, itself as an instance of CardAgent and some of its own unconscious entities, as well as its own collection of Xidea instances.

Basic behavioural cycle of a simple CardAgent in terms of the model of attention and memory

In a reworking of CardWorld1(a) the CardAgent will have a single main goal of satisfying the CardAgent's ideas of the other (i.e. human) CardAgent's requests or questions. Subgoals will correspond to abilities of the CardAgent: to move and turn PlayingCards, and to move, turn, shuffle, spread, and stack CardPiles. Subgoals will be pursued, leading to actions, until the human CardAgent's requests or questions, as understood by the programmed CardAgent, have been satisfied.

Attention, focus, and memory

Between turns of the agents involved in a dialogue, the CardAgent will have a set of Xidea representations of

- what events occurred in the most recent turn;
- the agent who caused those events;
- which of its own goals (if any) were accomplished by those events;
- which of the other agent's goals were (presumably) accomplished by those events;
- the current state of the world and the state just before the most recent turn;
- its currently unaccomplished or continuing goals;

The actual contents of *attention*, limited to 8 or 9 items at a time, will be determined by an active process called the **noticer**, which will build Xidea representations based on a (potentially dynamic) priority queue of *types of things to be noticed*. It might include, for instance: events, agents, goals, cards, piles, locations, spreadness, stackedness, quantities, colors, suits, ranks, overlaps, nearness, ... Values noticed will generally be from vague to specific – e.g., quantity from: {1, 2, 3, ... , 7, several, many} first, then more precise numbers if time and computational resources permit before a higher-priority thing to be noticed (such as a new event) intervenes.

Memory might consist of a chronological sequence of Xideas. Each individual Xidea instance would be linked to the most recent state of attention to which it belongs. If it still corresponded to the current state of the world, the link might be updated, if the Xidea remained in attention. If it no longer corresponded to the state of the world, it would become an **Xmemory**, which could still remain in attention, for instance if it were brought to attention by an utterance. Perhaps better would be to create a new Xidea to correspond to an Xmemory which is brought back to attention. The new Xidea's properties might be only a proper subset of the properties of the Xmemory; that could model memory degradation.

Problem: How should continuity of object identity be tracked over time? Cards, for instance, should retain their identities after being moved or turned over. Should piles of cards retain their identities until they disappear by removal of cards or are combined with other piles? Probably there needs to be a distinction between **permanent** objects and **temporary** objects.

Enhanced model of a CardPlayer's behavioural cycle

Using the idea of a **noticer**, the behaviour of a CardPlayer on each turn can be expanded as follows:

1. Have the **noticer**, using its priority queue, make a list of **Xideas** for *things noticed* since the last turn (and some possibly remaining from the previous list of things noticed), in order of priority to be dealt with. For example, events that have occurred since the previous turn will likely be near the front of that list.
2. Create goals for dealing with one or more of the things noticed.
3. Take one (or more) goal(s) from the top of the goal priority queue and act on it, putting a corresponding eventIdea into the priority queue of things noticed, along with a reference to the goal that triggered the event.
4. Repeat step 3 until some appropriate utterance event has been made to the other agent, signalling the end of the turn.

Xideas in a representation of the current situation

There should be Xideas for various levels of abstraction in the representation of the current situation. For example, a particular playing card like the Ace of Spades can be represented as

- a **type**, with suit Spade and rank Ace, but without location or face up/down on the current card table,
- a **token**, present at the current card table, with other properties (whose values may be currently unknown): location, face up/down, owner, membership in a pile, etc. A token may or may not be *unique* in the current situation.

Relationships between Xideas can involve (among many others) *is-part-of*, *has-parts*, *belongs-to*, *moved*, *was moved by*, *turned*, *was-turned-by*, *shuffled*, *was-shuffled-by*, *stacked*, *was-stacked-by*, etc. Those relationships, in turn, should be represented as Xideas of both *type* and *token*.

More primitive still than a token is a **notice** or **observation** of a token. Those are what the noticer creates, and they are the basis of memory. An immediate problem for a CardAgent is to relate each notice to a token in the current context, either a new token or a token corresponding to a previous notice. For example, if a CardAgent knows that a particular playing card is unique in the current context, then a notice of the Ace of Spades always corresponds to the same playing card token. However, if it is possible that there are two or more Aces of Spades in the current context, then a new notice of an Ace of Spades may or may not correspond to the same token to which a previous notice of Ace of Spades corresponded. Smooth, continuous movements (represented by **event token notices**) normally preserve token identity, but abrupt jumps of card notices, in a situation of non-unique cards, might correspond to different card identities.

Framing the situation

A **frame** is an Xidea type for representing a particular kind of situation. Frames provide a means for representing long-term memories and for initiating Xideas in a situation. For example, if a CardWorld situation is framed as involving a single standard deck of playing cards, then that belief can instantiate, among other things, 52 PlayingCardIdea instances, each representing one of the 52 unique cards in the deck, plus an unknown number (0, 1, or 2) of jokers. That might lead to a goal to determine the exact number of jokers, whether by observation or by questioning the other agent – assuming that the situation frame involves another agent. Xideas provided by a frame, but which have not yet been confirmed in the current situation, will be called **expectations**.

Discrepancies observed in the situation from values in the frame's *expectations* may give rise to goals to resolve the discrepancies. For example, if two Ace of Spades observations are made simultaneously, that would conflict with the assumption of a single standard deck of cards.

A frame *token* represents a particular instance of the situation, while a frame *observation* (or notice) represents an observed *state* of the situation. Short-term memories for states of the situation consist of a doubly-linked chain of frame observations of it. As the situation changes, some of the frame observations may be dropped from the short-term memory chain. In the CardWorld, frame observations, in turn, will consist of observations of CardPiles, PlayingCards, Agents, Events, etc.

Parts of a frame instance that don't change from one frame state observation to the next need not be duplicated.

Frames are like acts and scenes in a play. Frame tokens are like *performances* of acts and scenes, and frame token *observations* are like an audience's watching the acts and scenes, although members of the audience can also be actors in the scenes. Frames can be nested to three or more levels. Fixed elements in a frame provide the context for *events in time* in the frame. *Episodic memory* is for events in a frame observation and for chains of frame observations.

Unlike a play, frames in normal life are incompletely scripted. That is, the action is improvisational.

Frames and scenes in CardWorld2

In CardWorld1 there are no explicit frames, no explicit goals, and no segmentation of action and memory into scenes. In CardWorld2 some of those things will be made explicit.

The **explicit frame** for CardWorld2 will consist of a standard deck of playing cards and two card agents, one human and one software. The software card agent will have an explicit goal to satisfy requests and answer questions from the human agent. It will also have a goal to acquire and remember information about the identities and locations of cards on the card table. The assumption of a standard deck of playing cards will provide expectations for the software agent, as well as for the human player, of course.

Actions in CardWorld2 can be segmented naturally into scenes by operations that shuffle and/or stack all of the cards in the deck. Short-term memory for events in the previous scene can be cleared when a new scene begins.

In CardWorld2 the human agent need not have any goals known to the software agent other than the goals to have his or her requests satisfied and questions answered. The human can just move cards and piles around, turn them over, and make requests to the software agent to shuffle, stack, or turn piles or to answer questions about cards and piles, and their identities, quantities, and locations. In later versions of CardWorld we can introduce other explicit goals, such as goals associated with particular games.