

Representing Multi-agent Plans Using Statecharts with Explicit Aggregation¹

Xiaohong Wu

MS Student, Dept. of Computer Science and NASA ACE
North Carolina A&T State University, Greensboro, NC 27411

Xiaohong@ncat.edu

Dr. Albert Esterline, Advisor

Abstract

We show how to represent multi-agent plans using the statechart formalism. We emphasize an explicit approach for representing aggregate plans. The assignment of agents to roles in a plan and agents as position holders are also discussed. Finally, our work is related to philosophical societal theories.

1. Introduction

We represent multi-agent plans with statecharts, which represent both hierarchies of states (hence abstraction hierarchies) and orthogonal (concurrent) components. Synchronization among components could be achieved implicitly, by modeling the aggregate through a web of references. Instead, we use an explicit approach, which makes plans more reusable and extendible. Section 2 introduces the statechart notation; Section 3 shows how statecharts are used to represent multi-agent plans. Section 4 describes the explicit approach to synchronizing aggregate plan. Subplans are not the agents but rather the role positions to which agents are assigned. Section 5 describes agents as position holders. Section 6 sketches how our approach relates to central notions in philosophical societal theory, such as joint intention, agreement, and mutual belief.

2. Statecharts

Statecharts [Har87] are a visual formalism to specify the behavior of complex reactive systems. They extend conventional state-transition diagrams to include the notions of hierarchy, concurrency (orthogonality), and communication. Statecharts also may be viewed at varied levels of detail through abstraction and refinement. Thus,

Statecharts = state-diagrams + depth + orthogonality + broadcast-communication

Hierarchy in statecharts is achieved by grouping states into superstates. This is a *XOR* (exclusive-or) decomposition, as shown in Figure 1. This statechart includes two states, A and B. A has two substates, C and D. If the system is in A, then it must be either in substate C or D, but not both (exclusive-or relation). The arrow indicates the transition direction; the label over it indicates the transition event. Suppose that the system is in state B. When event *e* occurs, it goes directly to substate C. When event *h* occurs, it goes to A as a whole (the arrow labeled "*h*" stops at the boundary of A). The unlabeled dotted arrow to substate D of A indicates that D is the default state of A. So, the system goes to D on event *h*. If when event *f* occurs, whatever substate of A the system is in, it goes to B (the arrow labeled "*f*" starts at the boundary of A). Finally, if the system is in substate C, when event *g* occurs, it goes to D if and only if condition *c* holds (see the "*c*" in square bracket after the label "*g*")

¹ This research was supported by grant NAG 2-1150, "Motion Planning in a Society of Intelligent Mobile Agents," from NASA Ames Research Center

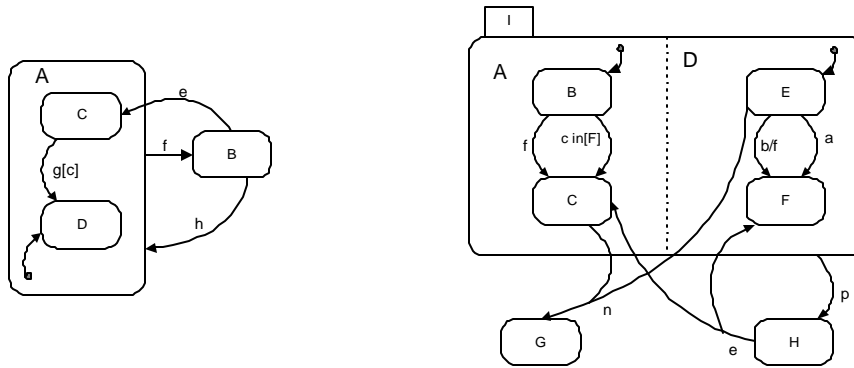


Figure 1 Statechart with XOR-state

Figure 2 Statechart with AND-state

Concurrency in statecharts is shown through orthogonality, which is an AND decomposition captured by the partition of the statechart as in Figure 2. There are three states, G, H and I. I has two orthogonal components, A and D (see the dashed boundary). When the system is in I, it must be in A and D. A and D also have XOR decompositions so that, if the system is in I, it must be in one of four possible pairs of substates $\langle B, E \rangle$, $\langle B, F \rangle$, $\langle C, E \rangle$ or $\langle C, F \rangle$. If the system is in the state H, when event e occurs, it enters I and ends up at substates C of A and F of D. If the system is in substate E of D and either B or C of A, then, when event a occurs, it goes to F and stays in whatever substate of A it was in. However, when event b occurs, a transition labeled b/f is taken, the action f is immediately activated, and is regarded as an (internal) event. Now, if the system is also in the substate B of A, then it goes to C; thus, the system goes to F and C when event b occurs. If the system is in substates C and E in I, when event n occurs, it leaves the state I and enters state G. The arrow labeled “ p ” indicates that, if the system is in any one of four pairs of substates of I aforementioned, when p occurs, it goes to state H.

The communication mechanism in statecharts is based on broadcast, whereby the sender may proceed even if nobody is listening. It also forces all enabled listeners to accept and respond to the message simultaneously.

3. Plans Represented by Statecharts

We give an example that represents with a statechart a plan for planting a plant. Figure 3 illustrates the dependency relations among the following steps: **1.** Dig hole. **2.** Put some soil back into the hole. **3.** Mix soil conditioner with the soil in the hole. **4.** Mix soil conditioner with the remaining soil out of the hole.

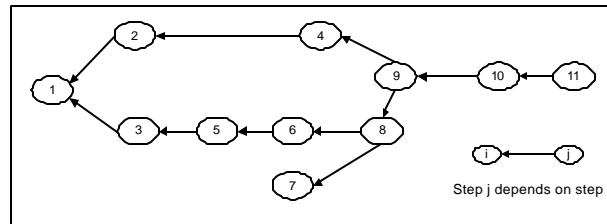


Figure 3 Dependencies Involved in Planting a Plant

5. Check planting depth; fill to planting depth. **6.** Pour in water with dissolved fertilizer. **7.** Take the plant out of the pot. **8.** Put the plant into the hole. **9.** Put the rest of the soil in the hole around the plant. **10.** Pour the rest of the water/fertilizer around the plant. **11.** Put mulch around the plant.

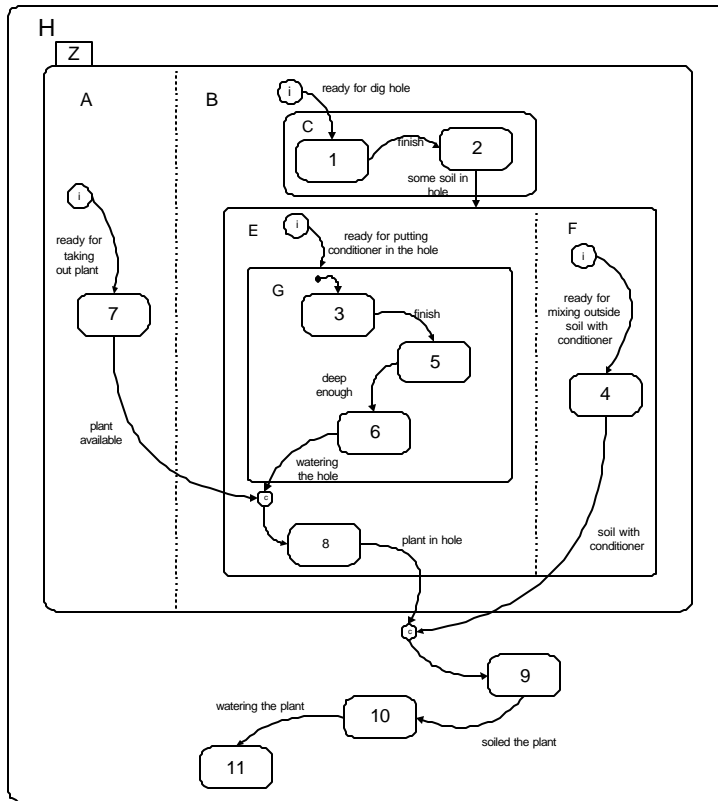


Figure 4 Plan-Statechart for Planting A Plant

Concurrency among the plans in a plan-statechart can be shown clearly and flexibly. For example, plan Z in Figure 4 has two orthogonal subplans, A and B, which can proceed in parallel except that subplan 7 of A must be carried out before the subplan 8 of B. Because we introduce the idle state, when the subplan 7 of A occurs is not constrained related to C, G, and F – so maximum concurrency is portrayed.

4. Whole-Parts Charts for Concurrency and Synchronization

Traditionally, interactions among orthogonal components are synchronized implicitly, that is, different components of an *AND*-state synchronize with each other by the combined effects of the broadcast and triggering mechanisms [PL97]. These implicit methods lack reusability, understandability and extendibility because implicit synchronization does not encapsulate the details of components of an *AND*-state – with any change in a part, the other parts and the *AND*-state as a whole must change. And, with any modification of the whole, the different parts must be modified. [PL97] proposed an explicit method that extracts synchronization aspects from the statechart and encapsulates them into an additional statechart, representing the whole. A *whole* represents a structural and behavioral abstraction inferred from associative knowledge in the domain [PL97]. The *whole* section and *parts* sections evolve concurrently as in an *AND*-state. The following communication constraints enforce self-containment and encapsulation: (1) Events cannot trespass the boundary among the different *parts* sections. (2) Events can be broadcast both from the *whole* section to the *parts* sections and vice versa. So the *whole* knows its *parts* but the *parts* ignore each other.

From this we derive a plan represented by the statechart shown in Figure 4. This statechart is a *plan-statechart*. We introduce an idle state in our plan-statecharts, a circle with an *i* inside. Whenever a state with an idle substate is entered, the idle state is a default state. We can decompose a plan into several smaller subplans carried out sequentially (*XOR*) or concurrently (*AND*). We also can group plans into a superplan hiding details. A circle with a *c* inside indicates joint conditions for a transition. In Figure 4, we omit the action part in the label of a transition, since it is simply the inception of the ensuing activity.

Synchronization aspects can be expressed entirely in the *whole* section without affecting any of the original behavioral descriptions of the orthogonal parts. The *whole* thus nicely caters for reusability, understandability and extensibility.

In plan-statecharts, we adopt the explicit method to show synchronization. A *whole-parts* chart is associated with any *AND*-plan involving interactions among its orthogonal components. A whole-parts chart is a rectangle with two sections, a parts section and a whole section, as illustrated in Figure 5. The interacting components are explicitly expressed as parts in a whole-parts chart as round-cornered rectangles separated by solid lines to emphasize rule (1). The events which control the synchronization among the parts are listed in the parts. The whole component is an additional entity to show explicitly how the parts are synchronized. A dash line separates the whole component from the parts to emphasize rule (2). A solid circle in the whole component indicates the start of the whole, and a solid circle within a circle indicates the termination of the whole.

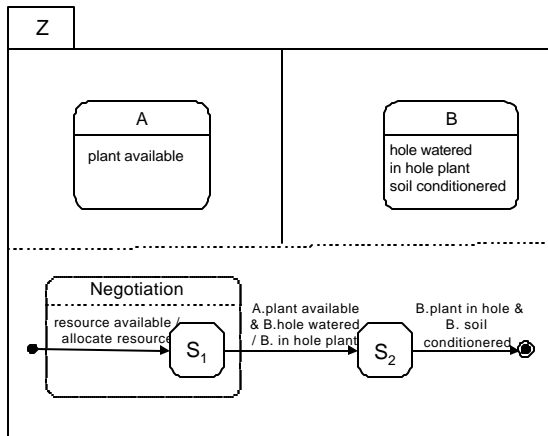


Figure 5 The Whole-Parts Chart of An AND-Plan Z

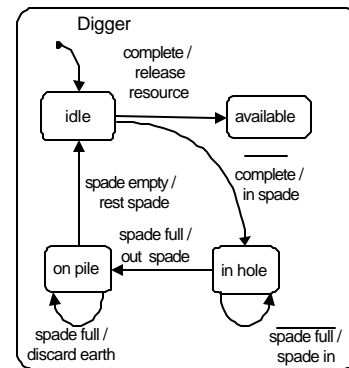


Figure 8 The Position-related Plan-Statechart for Digger

Figure 5 is the whole-parts chart for *AND*-plan Z in the figure 4. Two orthogonal components, A and B, of Z are two parts in this chart. The synchronizing events are listed in the parts. A prefix notation indicates the source of events, so that, for example, *A.plant available* means the *plant available* event comes from plan A. The *Negotiation* part in the whole component shows how to allocate the resources. The Z plan as a whole will advance from state S_1 to S_2 if *A.plant available* and *B.hole watered* are true. This transition also initializes the inception of S_2 's activity *B.in hole plant*. The occurrence of events *B.plant in hole* and *B.soil conditioned* will lead to the exit from Z.

5. Agents as Position Holders in Plan-Statecharts

When a plan is decomposed at a deep level, the subplans show plans for role positions, not agents. This allows flexibility in the agent assignment process, since, whoever in the qualification set is assigned to a position, he, as a position holder, follows the role plan. The agents involved in a plan form a group with structure as discussed in societal theory [Tu95]. For example, the *dig-hole* plan in Figure 4 (subplan1) is an *AND*-plan with two parts, diggers and a hole. At this level, how the actions are to be performed is shown.

Therefore, we use the position-related plan-statecharts to show the position holder's plan. Figures 6 and 7 are the plan-statecharts for the *digger1* and *hole* position holders.

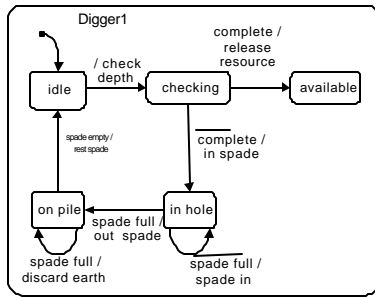


Figure 6 The Position-Related Plan-Statecharts for Digger1's Plan

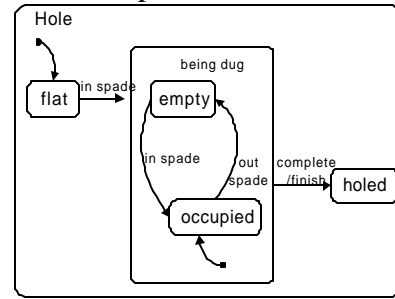


Figure 7 The Position-Related Plan-Statechart fro Hole's Plan

The *dig-hole* plan may involve more than one digger. The other diggers' positions (called *digger*) differ from the position *Digger1* by lacking a checking step. Figure 8 shows the general digger's position plan. When several diggers are assigned to the *dig-hole* plan, the actions among the diggers must be synchronized to avoid their spades colliding. Figure 9 shows two diggers' whole-parts chart for *dig-hole*. The negotiation part in section *whole_dig-hole* assigns two agents to the positions *digger1* and *digger* and initiates the plan. The collisions are extracted and modeled in the *whole* section, which explicitly expresses the synchronization among the components.

Explicitly expressing the synchronization aspects entirely in a *whole* section provides a natural context for extendibility. We can add one more digger in the *dig-hole* plan without any change to the parts already present and by including just one additional state in the whole – clearly showing the reusability of plans.

6. Societal Theory for Multi-Agent Systems

The bridge between a plan-statechart and a multi-agent system is the negotiation process in the whole-parts chart associated with every *AND*-plan. Agents in a multi-agent system use negotiation for conflict resolution and coordination [Mül 96]. The negotiation in a

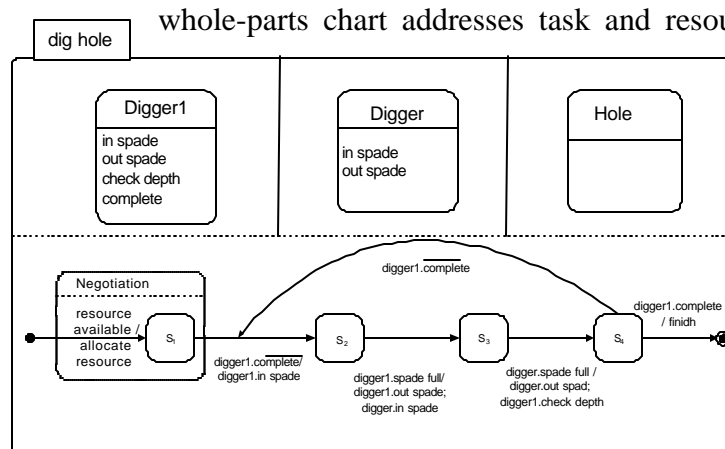


Figure 9 The Whole-Parts Chart for Dig-Hole Plan

whole-parts chart addresses task and resource allocation, since our model assumes that the conflicts among the position holders is already resolved. Based on societal theory, agents involved in a plan can be viewed as a social group with task and role structure, and this structure is characterized by social norms. A plan is formed based on agreements within an authority system.

The plan consists of the positions, which need to be

filled by the executive agents, the subplans or tasks, which the position holders should carry out, and the rules, which govern tasks and duties that agents have in their positions. When the agents in a social group fill the positions in a plan, they not only obtain knowledge about their own subplans and the rules, but also know that other agents know their subplans and rules. When an agent agrees to be a position holder, he also agrees to follow his subplan and the rules. Therefore, the agents, as position holders of a plan, not only form a joint intention but also an agreement to jointly perform the plan. Furthermore, the agents must mutually believe that they made such an agreement, and the agents must commit to doing their parts of the job. Mutual belief is like common knowledge [FHW95] in a group except that it is generally future directed.

After the negotiation process in the *whole* section of a whole-parts chart, the bridge between a multi-agent plan and a multi-agent system is established. The joint intention formed in the negotiation process can serve to initiate joint performance for a plan, and the agreement will guide the agents' behavior and effect coordination.

7. Conclusion

We have shown how statecharts are used to represent multi-agent plans and have emphasized an explicit approach to modeling an aggregate plan, explicitly representing the whole plan. This approach makes plans more reusable and extensible. It is important that agents are not represented by parts of statecharts but rather are assigned such parts. For one thing, this allows a useful notion of the state of an agent, namely, a pair consisting of the history of the agent and a set pointers to the parts of his plan currently being executed. The hierarchical nature of statecharts allows an agent's history to be summarized by remembering only higher-level states when details are not necessary. Given this notion of a state, a modal analysis of multi-agent systems [FHM95] is possible. In particular, the key notion of common knowledge is available, and, indeed, our analysis supplies structuring for groups and histories that enhances the application of this notion. Our statechart-based approach also supports a deontic analysis of agent actions in that transitions can be seen as obligated actions. A plan-statechart could be extended with violation transitions and states, which are thus forbidden but may actually occur – allowing for fault analysis. Finally, these modal notions lead into the larger topic of societal theory, only touched on in this paper

Reference

- [Har87] Harel, David., "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming* 8 (1987), pp. 231-274.
- [FHM95] Fagin, R., Halpun, J. Y., Moses, Y., Vardi, M. Y., *Reasoning about Knowledge*, Cambridge, MA: The MIT Press, 1995.
- [Mül 96] Müller, J. H. "Negotiation Principles." in G.M.P. O'Hare and N.R. Jennings (eds), *Foundations of Distributed Artificial Intelligence*, New York: Wiley, 1996, pp. 211-229.
- [PL97] Pazzi, L., "Extending statecharts for representin parts and wholes." *Proceedings of the 23rd EUROMICRO Conference*, 1997.
- [Tuo95] Tuomela, Raimo., *The Improtance of Us: A Philosophical Study of Basic Social Notions*, Stanford, CA: Standford University Press, 1995.