Conclusions & Future Work 12.

In this last chapter we will give a summary of the contributions of this work and present proposals for extending Ymir. Specifically, we will look at the following issues: {1} contributions of this thesis and what remains to be done {2}, potential limitations of the architecture, and {3} how Ymir could support extensions such as dialogue history, simulated emotions, learning and multiple conversant situations.

12.1 High-Level Issues

Two high-level issues were to be addressed by this work. The first was the general issue human-computer interaction. The plan was to make an interface that takes advantage of people's knowledge about face-to-face interaction, turn-taking and perceptual abilities of interacting parties to provide a consistent metaphor for the relationship between human and computer. The prototype designed shows that these features can, in fact, be incorporated into a computer interface, and that the outcome gives its user a high-bandwidth communication channel with the computer. Expansion of the prototype presented here will result in robust agent-based systems that provide a powerful and intuitive new means for people to interact with computers. Potential applications are simply too numerous to list, covering such broad areas as education, office work (and work in general), entertainment and psychological and social research.

The second general issue addressed in this thesis was dialogue modeling. The Ymir architecture is a model of face-to-face communication, linking together a number of elements of multimodal dialogue for the purpose of providing a platform for creating communicative humanoids. Gandalf, the first prototype built in Ymir, has been provided with a set of skills that enabled him to carry on multimodal dialogue and exhibit behavior comparable to that of a human engaged in dialogue. That it does exhibit behavior similar to a human conversational partner has been shown both by user testing and by a comparison to the Model Human Processor [Card et al. 1983]. Thus, Ymir has proven too be a sufficiently sophisticated architecture to provide the basis for designing interactive agents with language and multimodal capability.

Now lets look at the specific issues addressed in this thesis and how they were answered.

12.2 The Goals of Bridging Between Sensory Input and Action Generation

A main argument in this thesis has been that face-to-face dialogue happens on *multiple levels of granularity*, both in perception and action ("Challenges of Real-Time Multimodal Dialogue" on page 66). It is no surprise, therefore, that the model proposed here as a framework for multimodal agents, Ymir, is multilayered (Figure 7-10 on page 107). It uses the concept of layers (see "Layers" on page 92) to cluster processes that share similar time-constraints. Thus, highly reactive perceptual and action processes are contained in one layer ("Reactive Layer (RL)" on page 93), more reflective processes share another layer ("Process Control Layer (PCL)" on page 93) and the least time-dependent actions have their own layer ("Content Layer (CL)" on page 95). A fourth layer hosts the agent's ability to gracefully integrate multiple action goals into coherent outward behavior ("Action Scheduler (AS)" on page 94). Thus, as far as answering how to bridge from sensory input to action generation, Ymir has proven successful.

12.2.1 Continuous Input and Output Over Multiple Modes

The issue of continuity in multimodal interaction is a complex one. People manage to perceive multimodal acts to an extraordinary degree [Goodwin 1981], but entangling the rules how they do it has not been easy. The problem contains at least two parts: Segmenting out the significant parts of a unimodal act, and recognizing the function of multimodal acts in real time. In Chapter 5. ("Functional Analysis: A Precursor to Content Interpretation and (sometimes) Feedback Generation" on page 71) we argued that the latter was needed in order to do the former properly. While the former has gotten some attention in the literature, the latter has not been addressed, and was therefore made the focus in this work. The solution to the problem of continuous input advanced here lies in treating events at multiple levels of detail and integrating them in a way that supports directly the actions needed for the dialogue interaction. Two kinds of processes were designed, *unimodal* and *multimodal*, the former providing data needed in the latter to determine the function of multimodal acts (see "Virtual Sensors" on page 98 and "Multimodal Descriptors" on page 99). Taking a simulated parallel approach, this model is likely to deal successfully with expansion—and thus the important issue of real-time constraints—should future implementation rely on massively parallel hardware.

The mirror problem to continuous input is continuous multimodal *output*. One part is the *coordination* of multimodal output, the other is *generation* of multimodal acts. The former, falling directly under the auspices of dialogue management, was the main focus here. Separating *coordination* from *generation* in this manner allows us to create direct links between the perceptual acts of a multimodal character and its actions, independent of the action's form. Decisions to act are made by a set of modules that monitor the conditions in the agent's perceptual and "cognitive" states ("State Decision Modules" on page 118). The actions of these modules are the *intentional phase* of a multimodal act; they in turn get evaluated in the Action Scheduler ("Action Scheduler (AS)" on page 94), which, depending on the current state of the agent's memory and motor state, turns them into appropriate motor actions.

12.2.2 Coordination of Actions on Multiple Levels

Having proposed a highly distributed system, we need a way for modules to speak to one another. For this purpose, Ymir uses three blackboards. For example, to enable separation between output generation and output *coordination*—or process control—the PCL and CL communicate with each other through a dedicated blackboard ("Blackboards" on page 96) using special communication primitives (Figure 8-8 on page 119). Such "limited access" blackboards allow us to accommodate multiple modules at multiple levels of granularity without sacrificing system comprehensibility.

Looking at Ymir at a more detailed level, the State Decision modules in the Process Control Layer (see "State Decision Modules" on page 118) can be made to turn on or off modules in that same layer, as well as in the layer below, the Reactive Layer. This is another promising method for exerting run-time control over multiple modules at multiple levels. However, this scheme might need to be expanded to deal with more complex perceptual modules.

12.2.3 Lack of Behaviors

One problem we encountered in our discussion of autonomous agents was the amount/number of behaviors one can put into the system ("The Lack of Behaviors Problem" on page 85). Ymir solves this problem by allowing hierarchical definitions of behaviors and enabling other parts of the system to access these at any level (“Representation of Behaviors” on page 100). The main limitation of this scheme, as embodied in the implementation, is the lack of composition rules for piecing together various parts of various behaviors, to allow for the emergence of new behaviors (see “Inherent Limitations” below).

12.2.4 The Natural Language Problem

As demonstrated in the J. Jr. prototype (“The Reactive-Reflective Integration Problem” on page 85), adding natural language capabilities to agents based on most of the current reactive-only architectures is problematic. In Ymir the layer containing the agent’s knowledge base(s), Topic Knowledge Base (TKB) and Dialogue Knowledge Base (DKB), is separated from the layers exerting control over when actions happen. This allows for seamless integration of high-level behaviors such as language with lower-level ones.

While the separation of TKB and DKB from the rest of the system deals successfully with the issue of integrating natural language into a hybrid reactive-reflective system, we still need to define the mechanisms that interpret and generate the linguistic output. The multimodal interpreter in the Gandalf prototype is very simple, and works well in its limited arena, but it leaves several issues unanswered. At the top of this list is how to deal with *spatial data* in relation to language. Only considerable additions to what has been implemented in the prototype can clarify how extensible this scheme is.

12.2.5 The Expansion Problem

In the J. Jr. chapter (page 81) we saw how expanding the agent’s behaviors and perceptual capabilities became a problem of system complexity. This is a general problem for systems based on finite state machines, and other systems which use mechanisms at the same level of granularity. Because Ymir separates reactive processes from reflective ones, expanding either one is possible without interfering with the other. Using communication between the Process Control Layer and the Content Layer (see Figure 8-8 on page 119) allows us to continuously expand the system’s topic knowledge without having to change a single thing regarding the real-time execution of actions.

One question that remains unanswered is how well the idea of hierarchical perception works. More specifically, can we add more advanced perceptual modules into the Process Control Layer and the Content Layer—does this make sense? In the current implementation these are mainly contained in the Reactive Layer. More research is needed.

12.2.6 Goals: Conclusion

In conclusion, Ymir provides a foundation that fulfills the original goals it was meant to fulfill: It bridges between multimodal input analysis and multimodal output generation, allowing the construction of autonomous humanoid agents capable of continuous, interactive dialogue with people, comparable to human face-to-face conversation, proposing a number of mechanisms and ideas to fill in the missing knowledge that could make this possible.

12.3 Inherent Limitations

While Ymir provides quite a number of solutions to generating communicative humanoids, it also is obvious that as an architecture for accommodating *everything* needed in multimodal dialogue, Ymir has its limitations. What exactly those are is not clear at the present; the hope is that it will provide at least a solid stepping stone to the next level. However, it may be useful for other researchers to provide an insight into where the weak links may lie.

12.3.1 Reactive-Reflective Distinction

One can guess that the strict distinction between reactive and reflective behaviors proposed may need to be relaxed, perhaps to the point where the division into Reactive and Process Control layers no longer make sense. Such a course of events, however, is likely to make the construction of complex characters quite daunting, and may call for new principles for conceptually segmenting the systems responsible for those functions. Schemes such as those used by Blumberg & Gaylean [1995], which allow modules to be grouped, could be a potential solution in this case. Still, the level at which the modules are designed is likely to be adequate for constructing even relatively complex agents.

12.3.2 Communication Between Layers

It may also happen that communication between the Process Control Layer and the Content Layer eventually becomes so complex that they should be considered a single system, or a larger set of smaller systems. Again, mechanisms must be introduced to steer away from increasing numbers of identical modules, messages or rules, since designing these becomes difficult.

12.3.3 Behaviors and Action Generation

A limitation of the current implementation of Ymir relates to the way behaviors are generated. Since all behaviors are named (with the exception of spatial values being inserted where needed), a programmer needs to design hundreds of behaviors to create a fully capable character. A better solution would be to create a motor representation scheme where more flexibility exists in the generation of behaviors, such that new actions could be composed of old ones. This would also be a necessary change to accommodate motor learning.

Having looked closely at the main issues Ymir was meant to provide a solution to, we are now ready to delve into its limitations on a more global scale—how will more general behaviors be integrated into Ymir?

12.4 Extending Ymir/Gandalf

The Ymir architecture was designed over a period of about 3 years. In the process, many options were considered and many were rejected. The foundation for its design and design trade-offs was laid with the following goals:

1. Implementable by a single person,
2. incorporating—or allows for the extension to incorporate—as many features of face-to-face interaction as possible,
3. flexible, modifiable, and
4. real-time performance.

Rather than trying to get everything "right" the first time around, the system design was slanted toward *flexibility* in the architecture. There are therefore several levels of "entry" that a character designer could access the model presented, from the lowest level of Logic Net design, state modules and virtual sensors, up to the hybrid framework that the whole system relies on. Parallel to this is reuse of the LISP code written, i.e. the implementation itself: parts of this code could be used, for example the knowledge bases, motor scheduler or intonation analysis, or it could be used whole-sale, with other software modules entering the system at particular points.

At the low level, the most obvious place to start would be to experiment with various new modules. At this level one could add learning or self-regulation, for example by using "b-brain" modules [Minsky 1987] that modify the conditions in other modules. This could make the interaction more robust and the system more able to deal with varied conditions. One could also enter the system at a slightly higher level, adding new types of modules, keeping part or all of the current ones. At the highest level one could use the general architecture of the layers and blackboards—even the modules themselves—but make entirely new module mechanisms. These are all ways to improve what Ymir currently has to offer. But we also want to add new features such as emotion, mobility, multiple participants, etc. For this we have to stretch the platform to incorporate new mechanisms.

12.4.1 Where Are We Now? Current Status

Several advanced features have been shown in the Gandalf prototype, among them deictic gesture recognition/reaction generation and single pair turns such as question->answer and request->action. Most of the current prototype's limitations come from the simplicity of the knowledge bases. Expansion of these would allow more features to be added. One such feature is multiple query-response turns: "Move the box / Do you mean the blue one? / No, sorry, move the sphere / This one? / Yes / Which way?", etc. Another feature that more extensive knowledge bases could allow are directives such as "Look at me" or "Look at me when I'm speaking". These are interesting language-based tools that help in the process of dialogue and may prove crucial for the success of future communicative agents.

Other features that could be added include emotional simulation, affecting behavior at many levels, linking gaze directly up with visual input and the spatial knowledge base, dialogue history, anaphora and other references to dialogue events, as well as recognition of more types of gestures and more extensive multimodal event representation. We would also want to extend Gandalf to be able to converse with more than one agent or person at a time. We will look at these in turn.

12.4.2 Multiple Turns for Single Utterances

To add multiple pair turns, one would need to create a topic knowledge base (TKB) with a parser that can mark the already-parsed utterances with the state of the dialogue. Such systems have already been built [Allen 1987] and we will not go into those here in any detail. The interface to this TKB would have to add a few primitives to the communication with the Process Control Layer (PCL). First, the PCL needs to recognize what class of utterance is being produced at any stage in the dialogue. This is needed to allow it to automatically add correct process control behaviors such as hesitations, paraverbals, even short verbal utterances such as "I know this...hang on a second" while gazing upward. Second, the PCL needs to send messages to the KB that don't exist in the current Gandalf prototype. These include telling it whether a certain utterance had been delivered, how it might have been modified, and what kinds of fillers, extraneous utterances and actions were taken. This kind of extension is relatively straightforward, and is currently being made.

12.4.3 Dialogue Process Directives

To enable a user to address an agent created in Ymir with directives such as “Look over there”, “Turn your head away”, “Stop staring!”, one needs special connections from the Dialogue Knowledge Base (DKB) to the Action Scheduler (AS) or the decision modules. A phrase like “Look at me” would produce an internal record, directed at a class of decision modules in the Process Control Layer, causing them to fire. A phrase like “Stop staring!” would produce a “constraints record” directed at a collection of behavior modules in the AS. These constraints would need to be fulfilled when the behaviors get executed, in the same manner that the motor composer trades off between behavior modules based on the time they take to execute. The action Look-at-User[1] (or the more advanced version Look-at-Person[X]) would have to fulfill the condition (not Look-at-User). Whenever activated, the process computing spatial location for the user would have to modify the value to not coincide with the user’s location. How extensible this scheme is remains to be seen, but it seems certain to work for at least a substantial number of simple cases.

12.4.4 Emotional Simulation

Emotions affect behavior at many levels, although reactive behaviors may show less effect than medium- to long-term planning ones. In keeping with the modular approach to character building, an emotional module could reach in to both the decision modules, affecting variables such as effective lifetimes and responsivity to conditions. It could also change the behavior modules in certain, more powerful, ways, for example by limiting the number of options for executing certain behaviors, or changing the time scales of execution. This would require a substantially more powerful Action Scheduler, and additional features for the behavior modules, but would no require a change in the Ymir architecture.

12.4.5 Spatial Sensors & their Link to Spatial Knowledge

If we want the agent’s gaze to represent information collection on its part, the eyes have to be connected directly with the flow of visual input. This could be achieved simply by using cameras for eyes. Such a system would have to keep a spatial knowledge base updated every time the agent looked in a certain direction. Without going into the details of object recognition and maintaining the permanence of objects from moment to moment, such an extension would fit nicely into the model

1. See “Spatio-Motor Skills” on page 103.

proposed for spatial information storage accessible to both the Process Control Layer, Knowledge Layer and the Action Scheduler presented in the section "Spatio-Motor Skills" on page 103.

12.4.6 Dialogue History

In the current implementation of Gandalf the history of events simply piles up in the blackboards; there are no processes available for picking out events to make them available to the knowledge bases. This makes it impossible to refer to for example past dialogue events with expressions such as "...when you looked at me and said 'hello'". Obviously we need such mechanisms to provide a full system. The matter is mainly one of adding retrieval mechanisms to the knowledge bases; ones that would look in the various blackboards and "re-package" the accumulated events (such as `look-at-user`, `user-looking-at-me`, etc.) in a format that is directly accessible to the parsing mechanisms in the knowledge bases. This could also allow for references to past topic events such as "Go back to when the airplane was on the other side of the runway". Albeit not simple, this addition can build on knowledge base work already done in other systems [cf. Tanimoto 1987].

12.4.7 Advanced Gesture Recognition & Multimodal Event Representation

How do we determine what kind of gesture is being performed? Recognizing pointing gestures when that is the only kind possible is trivial in contrast to the situation where any gesture is possible—deictic, pantomimic, iconic, emblems, self-adjustors—and they all have to be discriminated in real-time. The solution proposed her is one where any kind of feature from any mode can support in deciding whether a certain behavior constitutes one type of gesture or another. For example, hearing the word "that" along with an extended arm might be enough of a trigger to classify the event as "deictic". Likewise, a verb combined with a complex hand motion and gaze that intersects hand position could be enough of a cue to signal an iconic gesture. Whatever may be a winning strategy for each kind of dialogue event, as argued in Chapter 5., the function of an event needs to be found before an action can be taken, and an action sometimes also needs to be taken before further processing can take place, as in the example of a deictic gesture where you *have* to look in the direction pointed to be able to follow the dialogue.

Along with the need to classify a number of manual gestures with different functions comes a need to package multimodal dialogue events so that they can be easily related to each other. Currently, speech is stored in frames (parse templates) where information about the words, type of utterance, type of dialogue event and intonation are stored. We need meta-frames that store combinations of multimodal events, perhaps at more than one level of detail. This would enable a knowledge base to access dialogue events in a coherent manner, similar to what people do when reviewing these events (a person does not store an event such as rising final intonation separately from the words spoken, but rather combines the two into packages called “questions”).

12.4.8 Multi-Participant Conversation

To allow Gandalf to participate in conversations with more than one user, or another agent, we need to first modify the perceptual modules particularly designed to detect events such as “looking-at-me” and “addressing-me” to use variables, where the variables can be assigned to the dialogue participants. This is not a radical addition to the architecture, but one that requires some additions in the dialogue knowledge base. These include coding of dialogue priority—currently the user has priority (and drives the interaction). If we want the agent to be able to take initiative, it needs to know when and how it is allowed to interrupt the user or other artificial agents. We might also have to look at the turn-taking mechanism to allow multi-participant information to influence turn behavior. Again, these are not major reworkings of the architecture—they can be built directly on the functionality that Ymir provides.

Whatever the future may hold for Ymir and Ymir-like architectures, its modular design makes it very likely to be useful for a number of years to come.