Advances in Cognitive Systems 7 (2019) 195–214 Submitted 8/2018; published 12/2019

Dialogue as Collaborative Problem Solving: A Case Study

James Allen JALLEN@IHMC.US
Choh Man Teng CMTENG@IHMC.US
Lucian Galescu LGALESCU@IHMC.US

Institute for Human and Machine Cognition, 40 South Alcaniz Street, Pensacola, FL 32502 USA

Abstract

A wide range of potential dialogue applications involve collaborative problem solving between humans and complex automated reasoning systems, but most existing dialogue system models use very simple dialogue models not expressive enough to capture such behavior. Existing models can only be employed for very simple tasks (e.g., making a reservation, querying bus schedules). We describe a model of dialogue explicitly based on a model of collaborative problem solving that can support quite complex task-based reasoning. The key contribution is a domain-independent collaborative problem-solving model that allows the user to interact with complex automated reasoning capabilities, including existing legacy systems, relatively easily. We describe this model and present a case study of the system running in a complex domain involving building and evaluating quantitative models of world processes (e.g., agriculture, economics, food security).

1. Introduction

Most current dialogue systems support only simple tasks that can be encoded in a state-based dialogue model, as in Williams et al. (2016) and the Dialogue State Tracking Challenge.[1] Approaches to dialogue modeling based on the information state (Cooper, 1997), such as TrindiKit (Larsson & Traum, 2000), its open-source successor trindikit.py (Ljunglöf, 2009), and OpenDial (Lison & Kennington, 2016), still support only very simple task and domain models. These are not expressive enough to support mixed-initiative dialogue systems with complex back-end reasoning systems.

Early theoretical work on SharedPlans (Grosz & Kraus, 1996; Lochbaum et al., 1990) and plan-based dialogue systems (e.g., Allen & Perrault, 1980; Litman & Allen, 1987) laid good foundations for more general systems. These defined the various communication acts that can be performed in terms of their effects on the dialogue participants' beliefs and goals. Perhaps the best-developed formalism is described by Cohen and Levesque (1990). Gabaldon et al. (2014) present an elaboration of such models within the ICARUS cognitive architecture. Allen et al. (2002) developed the concept of interpreting dialogue within an explicit model of collaborative problem solving (CPS), but while it inspired a number of subsequent systems, the CPS model has never been implemented in a direct way.

[1] https://www.microsoft.com/en-us/research/event/dialog-state-tracking-challenge

RavenClaw (Bohus & Rudnicky, 2009) supports a plan-based dialogue management framework that has been used in a number of dialogue systems. While its dialogue engine is task independent and includes a number of generic conversational skills, its behavior, from language interpretation to response generation, is driven by task-specific dialogue trees (the plans); these can describe only simple tasks and must be implemented anew for every application. The Disco system (Rich & Sidner, 2012) provides a more general framework tied to a plan-based representtation of the task. It requires specifying the domain plans as hierarchical task networks that are developed for the particular application. While these plan-based approaches allow more complexity in dialogues, they do not support conversations that involve collaborative interaction with complex back-end reasoners, such as planning and simulation engines, or complex software for data analysis.

The key contribution of our approach is that it uses a domain-independent model of collaborative problem solving and language understanding to support interaction, allowing independently developed complex reasoning components to implement the task. The first steps towards this are described in Galescu et al. (2018), which introduced a domain-independent model of collaborative problem-solving acts and a framework for deriving such acts based on domain-independent language understanding, intention recognition, and interaction with a domain-specific *behavioral agent* that manages the back-end reasoning capabilities and is built from scratch for each application. They reported a generic dialogue shell that can be used to produce a complete dialogue system with this framework.

This paper describes a further elaboration of this model in which the behavioral agent itself is also domain independent and captures an abstract model of collaborative problem solving that is easily instantiated with domain-specific reasoning components that can perform tasks such as planning, simulation, and causal analysis. While this behavioral agent is domain independent, it is still task dependent in that it is general across applications where the dialogue involves problem solving. We demonstrate this model in a complex domain, namely collaborative building, running, and evaluating models of world processes for purposes of making predictions and planning interventions, which we call *world modeling*.

1.1 Collaborative Problem Solving

When agents are engaged in solving problems together, they must communicate to agree on what goals to pursue, what steps to take to achieve those goals, negotiate roles, resources, and more. To underscore its collaborative aspect, this type of joint activity has been called Collaborative Problem Solving (CPS). Modeling the type of dialogues agents are engaged in during CPS must, therefore, take into account the nature of the joint activity itself. In the early 2000s, Allen and colleagues described a preliminary plan-based CPS model of dialogue based on an analysis of an agent's collaborative behavior at various levels:

- An **individual problem-solving level**, where each agent manages its own problem-solving state, and plans and executes individual actions;
- A **collaborative problem-solving level**, which models and manages the joint or collaborative problem-solving state (shared goals, resources, situations);
- An **interaction level**, where individual agents negotiate changes in joint problem-solving state;
- A **communication level**, where speech acts realize the interaction level acts.

This model was refined in a series of publications, and several prototype systems, starting from the TRIPS system (Allen et al., 2000), have been developed within this framework (Allen et al.,

2002; Blaylock & Allen, 2005; Allen et al., 2007; Ferguson & Allen, 2007). While the CPS model provided the theoretical foundations, in practice, each of these systems involved the use of a domain-specific CPS model built specifically for the application and encoded domain-specific heuristics in order to support intention recognition.

One of the main goals of our current work has been to enable linguistic interpretation and high-level intention recognition to be performed independently of the domain-specific problem-solving mechanisms. The domain-specific CPS would then specialize the higher-level intentions into concrete problem-solving actions and verify that such actions make sense in the domain context. As a consequence, in this model the back-end problem solvers would be relatively insulated from the need to worry about linguistic issues of sentence understanding, discourse, and dialogue management.

Galescu et al. (2018) describe a generic dialogue shell based on the CPS model that provides general language understanding and dialogue management capabilities. This dialogue shell has been used in a range of other systems, including a mixed-initiative system for planning and execution in a blocks world (Perera et al., 2017), learning about structures in blocks world (Perera et al., 2018), an assistant to a biologist for building, visualizing, running, and modifying complex biological causal models (Gyori et al., 2017), helping a human composer create and edit music scores (Quick & Morrison, 2017), and playing cooperative games (Kim et al., 2018). Each one of these systems uses very different forms of domain-specific reasoning, but all use the same interface to the generic dialogue shell. In this paper, we focus on pushing this model one step further by providing a generic behavioral agent performing the collaborative problem solving that could be used in different domains. This model is intended to support any application that that involves interpreting and analyzing situations, predicting outcomes, and planning possible interventions. We illustrate this model by showing its use in one very general setting, namely world modeling, where the system collaborates with the user to model various world processes, as described in the next section.

1.2 Case Study Domain: World Modeling

The state of the art in world modeling involves an extremely labor-intensive process, requiring person-years of effort by highly trained modelers in order to construct models and run experiments for analysis of complex problems such as food security, migration, and land use. The bulk of this effort does not actually involve constructing new quantitative models; rather, it is focused first on determining how a scenario corresponds to a configuration of quantitative modeling engines (including identifying or approximating the required data and parameter values needed), and then on the mechanics of running models over a set of scenario variations. For example, in building the Australian National Outlook (Hattfield-Dodds et al., 2015), over 50% of the effort was spent manually running simulations over multiple scenarios (e.g., manual linking of models, adjustment of parameters).

The existing state of the art requires human analysts to do the bulk of the work, identifying the capabilities of each modeling engine, selecting the relevant ones, examining historical data to determine possible ranges of values, harmonizing the data for input and connecting the output of some engines to other engines, possibly via some transformation, and configuring the whole network of modeling engines to perform the desired analyses. Often, solving a problem requires running a subset of models and caching the results in large data sets that are then accessed by the next stage of modeling. As a result, it is not feasible to construct such analyses in a timely manner to support the evaluation of options on ongoing problems critical to global security.

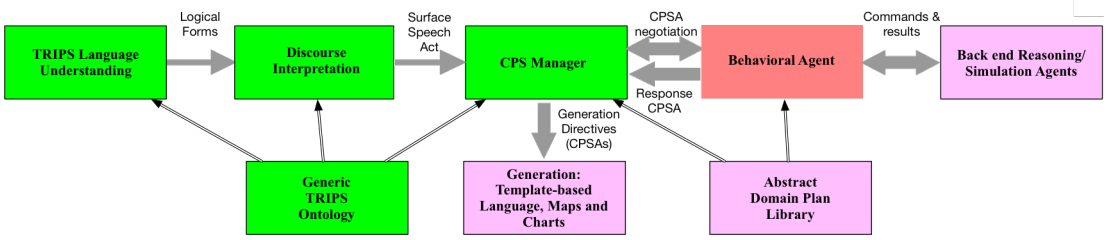

Figure 1. The abstract architecture showing domain and task general components (green), domain general components for problem-solving tasks (orange), and domain specific components (pink).

The long-term goal of the world modeling project is to develop a dialogue system that can provide analysts with a rich suite of tools with which they can quickly set up models to answer new questions, identify the necessary data, make the required assumptions, and then run complex experiments varying key parameters. This paper uses as the case study the first prototype of this system, called **CWMS (Collaborative World Modeling System**, pronounced "kooms"). In CWMS, the user and the system collaboratively explore different world modeling scenarios. The two communicate via dialogue in natural language and GUIs (e.g., visualization tools) to iteratively plan and refine simulation experiments together.

2. Dialogue Systems Based on Collaborative Problem Solving

A collaborative conversational agent must understand a user's utterances, that is, obtain a representation of the meaning of the utterance, recognize its intention, and then reason with this intention to decide what to do and/or say next. In addition, the system must convert its own intentions into language and communicate them to the user.

Figure 1 shows a conceptual diagram of our generic dialogue system. This follows the common separation of a conversational agent's functionality into *interpretation*, *behavior,* and *generation*, but where the separation lines are is critical for realizing the idea of isolating domain-independent from domain-specific processing. From the user's utterance the TRIPS parser (Allen & Teng, 2017; Allen et al., 2018) constructs a *logical form*, which is a domain-independent semantic representation. The logical forms are further processed into surface speech acts, including social conversational acts, by considering the discourse context and performing language-based intention recognition. Different surface forms are also normalized (see Section 4 for more details). The *CPS manager* is responsible for managing the collaborative problem-solving state (e.g., managing the proposals, acceptances, and rejections) required to reach agreement (mutual goals and beliefs). It converts the basic communicative acts into possible *abstract communicative intentions*. These intentions then are further evaluated with respect to the actual problem-solving state, so they are not fully interpreted until they are accepted by the *behavioral agent.* This agent is responsible for operationalizing the communicative intentions into actions (e.g., planning, acting on the world, updating its knowledge of the situation). An autonomous behavioral agent might be able to plan and act on its own, but neither the behavioral agent nor the user can unilaterally decide on the status of collaborative goals without a commitment from the other party. The two-way negotiation between the CPS manager and the behavioral agent includes communicating the behavioral agent's attitude towards shared goals,

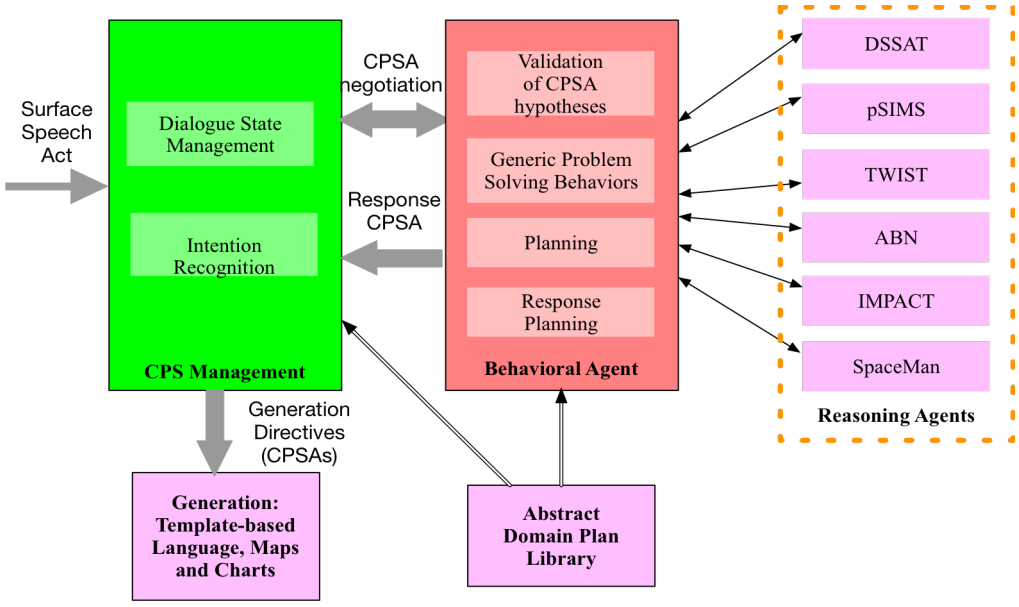

Figure 2. The context for the CWMS behavioral agent showing components that are generic (green), task specific but domain general (orange), and task and domain specific (pink).

which the CPS manager will use to generate communicative acts for language generation (such as accepting or rejecting a goal or proposing a new one) and update the collaborative state.

A more detailed view of the behavioral agent and back-end reasoners in the CWMS system is shown in Figure 2. As discussed above, the parsing, discourse interpretation, and collaborative problem-solving manager are mostly domain- and task-independent, but may use some domain-specific knowledge sources. The behavioral agent is task specific (i.e., analyzing situations and evaluating courses of action), though domain independent. In other words, it encodes general problem-solving behaviors such as planning, evaluating hypotheses, and acquiring and displaying information. It is designed to be able to work with any combination of domain-specific simulation and reasoning engines, shown in pink. Each simulation/reasoning module, as it comes online, sends a message registering its requirements (input) and capabilities (output). From these specifications, the planner will develop one or more viable plans linking the different modules to reach the desired simulation goal by feeding the outputs of particular modules (possibly with data transformations) as inputs to other modules.

We incorporated five domain specific simulation modules into our initial system:

1. **DSSAT** (Decision Support System for Agrotechnology Transfer; Jones et al., 2003): A point-based crop-modeling system with detailed models for over 40 crops.
2. **pSIMS** (parallel System for integrating Impact Models and Sectors; Elliott et al., 2014): A framework for massively parallel climate-impact simulations and global grid-based crop modeling, using as the underlying crop-modeling engine DSSAT (and others).

3. **ABN** (Agent Based Network; Marchand et al., 2016): An agent-based out-of-equilibrium food-shock model that computes changes in import, export, consumption, and reserves due to an injected shock to a country's crop production.
4. **TWIST** (Schewe et al., 2017): A short-term equilibrium model for computing market prices based on producer-side and consumer-side reserves, supply, demand, production, and consumption.
5. **IMPACT** (International Model for Policy Analysis of Agricultural Commodities and Trade; Robinson et al., 2015): A partial equilibrium multi-market economic model for analyzing long-term scenarios (30 years) involving agriculture, trade, and food security.

Table 1 shows an actual dialogue with the current system, which we will use to explore the system operation in the rest of the paper. As one can see, it starts with the user suggesting a goal (1), which the system accepts and refines (2), after which it suggests an initial course of action (4). To support this interaction, the system reasoned that malnourishment is an indicator of food insecurity and also that it is a value computable by one of the reasoning engines. The system then constructs a plan to run a reasoning engine that can compute expected childhood malnutrition rates, using baseline information such as expected availability of food. The answer computed is given in (6). The user then expresses a desire to elaborate the scenario based on predictions that the coming year will include an El Niño event. The system recognizes the intention to run a new analysis with this new assumption.

To estimate the amount of change, CWMS must construct a plan to estimate the change in crop yields given an El Niño year. A key problem in prediction is that the future weather and other factors are unknown. Thus, the system constructs a plan to estimate the effect of an El Niño year by interpolating results from simulations of 30 past years, weighting each result based on how much it was an El Niño year and its distance. Once the change in crop yield is computed, this change can then be fed through an economic model to compute the effect on food availability, which is then fed through a food security model that computes malnourishment rates.

The user follows up on the analysis with an open-ended question about what could be done to mitigate the effect of the El Niño event. CWMS does not have a specific answer. It does know a strategy to help the user explore causal connections, however, so in response it generates a causal graph and generates utterance (12). Using this information, the user asks about the effect of changing the planting date. The system knows a problem-solving strategy for exploring the effects of changing variables, and so asks whether it should construct a simulation experiment that estimates the crop yields for a range of different planting dates.

Once the user concurs, CWMS builds a plan for the experiment. Basically, it chooses a range of planting dates one week apart, starting one month before the typical planting date and running to one month after. For each of these dates, it estimates the expected yield using the interpolation function for El Niño operating over the 30 years of past data. Once completed, it shows a plot of the results to the user and identifies the best option. This is as far as we have space to discuss, but the user could easily continue on, asking for elaboration (e.g., *OK, and how would that affect the malnourishment rates?*), exploring other options (e.g., *What if we could increase the amount of fertilizer that was available?*), or pursuing some new strategy for dealing with the problem (e.g., shipping more food aid to the region).

Table 1. A dialogue between a user (U) and the CWMS system (S).

1	U:	*Can you analyze food insecurity in Sudan next year?*
2	S:	*Should we look at child malnourishment rates?*
3	U:	*OK.*
4	S:	*Should I compute a baseline estimate based on available data?*
5	U:	*yes*
		System identifies a reasoning engine (IMPACT) that can estimate childhood malnourishment, and uses existing knowledge about Sudan (population, food availability, etc.) to estimate the number of malnourished children.
6	S:	*OK. The percentage of malnourished children is about 39%.*
7	U:	*What if 2019 is an El Nino year?*
		System builds a plan to estimate the difference in crop yields in Sudan in El Nino years by interpolating simulation results from 30 years of prior weather data.
8	S:	*I can estimate the effect with the following plan. Does that look good?*
9	U:	*Yes*
10	S:	*OK. The percentage of malnourished children increases by 5%.*
11	U:	*What can we do to mitigate the increase?*
		System creates and displays a causal graph based on the knowledge it has acquired from the declarations of the reasoning engines.
12	S:	*Here is what I know about causal influences of malnourishment*
13	U:	*What would happen if we change the planting date?*
14	S:	*Shall I run simulations varying the planting date?*
15	U:	*Yes*
		System creates a plan to run a series of simulations that vary the planting date parameter, again estimating yields for an El Nino year by interpolating over results obtained from simulations of the past 30 years. The plan then assembles the results and presents to the user in a chart.
16	S:	*The best scenario involves planting crops one week earlier than usual.*

3. System Details

This section first discusses some aspects of the generic system applicable in any domain, and then considers the domain-specific components of the CWMS system. We use the dialogue in Table 1 as a running example, and elaborate on the components shown in Figure 2.

3.1 Language and Discourse Processing

While not the focus of this paper, to provide context this section describes briefly the language and discourse processing performed by the system. The TRIPS parser has an extensive grammar and an effective 100,000+ word semantic vocabulary defined in terms of a roughly 4000 concept ontology, described in more detail in Allen et al. (2018) and in Allen and Teng (2017). After a first pass that performs named-entity recognition, using geographical term databases, the parser produces a representation in a unscoped modal logic (see Allen, 1995; Manshadi et al., 2008). A fragment of the parse for the first sentence is shown in graphical form in Figure 3.

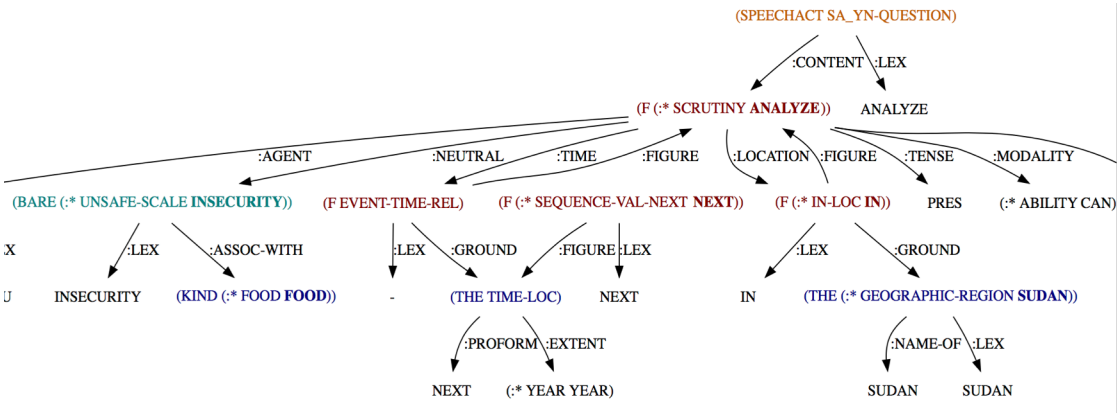

Figure 3. A fragment of the parse for "Can you analyze food insecurity in Sudan next year?".

The logical form is then passed to the interpretation manager, which performs a number of functions, most notably: (1) utterance segmentation; (2) surface speech act identification; (3) reference resolution; and (4) ontology mapping. Utterance segmentation handles situations where the user might perform multiple speech acts in a single utterance, such as an acceptance and an assertion (e.g., *Yes. I'll get the pizza*). Segmentation is accomplished by identifying various coordination structures in the logical form produced by the parser. The interpretation manager then proceeds to process each speech act individually, updating the discourse context after each one.

The second phase is identification of the surface speech act, which encodes the conventional ways that basic communicative intentions can be realized. The basic speech acts include assertions (TELL), commands (REQUEST), acknowledgements, acceptance and rejection acts, proposals, and various forms of questions and answers. It uses patterns based on convention linguistic forms to identify the range of likely speech acts. For example, the sentence *Can you open the door* would be identified most likely to be a REQUEST (to open the door), with a less likely option that it is a true yes/no question about the ability to open the door. This phase of processing provides a significant canonicalization for the range of possible ways speech acts can be realized. For instance, the following acts are all identified as asking a question about the time: *What is the time? Can you tell me the time? Do you know the time? Tell me the time. I want to know the time.* For the most part, these are all interpretations that could be derived from first principles using the plan-based model of speech acts (Allen & Perrault, 1980). With our current approach, however, these so-called indirect forms identified by inference in the plan-based model are derived directly and with no additional "cognitive effort", and thus is consistent with what we know of human processing from psycholinguistic research. The system has slightly less than one hundred hand-built rules that encode conventional common indirect forms found in conversation. Note the rules do not identify a single interpretation. Rather, they identify a ranked set of possible interpretations based on typical usage and the current discourse context. These options are passed to the next phase, which can then use additional context to identify the interpretation that was actually intended.

Let us return to our running example. The sentence *Can you analyze food insecurity in Sudan next year?*, while literally a yes-no question about ability, is mapped to a REQUEST act as first choice. Namely, it is a request that the system perform an analysis action (ONT::SCRUTINY) of the food insecurity situation in Sudan.

The next phase in the interpretation manager is reference resolution, in which referring expressions such as anaphora and definite descriptions are connected to previously mentioned terms in the discourse. As typical with the first utterance in a dialogue, there are no anaphoric expressions in our example sentence. Note that external name references have already been handled by the named-entity recognition component that was part of preprocessing. Thus, external names are already tagged with their referential information before parsing.

In many cases, the logical form produced by parsing is transformed into a new representation based on the domain ontology using an ontology mapping system. In systems where such mapping rules are defined, we can automatically compute which TRIPS ontology types correspond to domain-specific types and prefer these senses. This process results in the surface speech act that is the starting point in Figure 2.

3.2 Collaborative Problem-Solving Management

The output of the discourse interpretation phase is a speech-act hypothesis that is the input to the collaborative problem-solving management phase. The CPS manager is designed to interpret and drive the interactions that embody problem-solving negotiation between the user and the system (i.e., the interaction level in the above discussion). It is an evolution of the Collaborative Problem Solving (CPS) model described by Allen et al. (2002), with a particular focus on separating the domain-independent aspects from the domain-specific reasoning. It is mainly concerned with interpreting and driving the interaction that establishes and manages the collaborative problem state, by means of CPS acts such as proposing a goal, accepting or rejecting a proposal, proposing the refinement of a goal, or identifying a solution to a problem. One might immediately realize that this cannot be done accurately without reasoning about domain-specific intentions as well.

For instance, in our running example, the sentence *Can you analyze food insecurity in Sudan next year?*, which has been mapped to a surface speech act REQUEST, is deemed likely to be a PROPOSE of a new top-level goal (since no goal has been established so far, this being the first utterance). All this can be done based on the current problem-solving context (i.e., no goal has been agreed to yet) and the form of the surface speech act (i.e., a REQUEST). But this hypothesis cannot be confirmed without checking that 'analyzing food insecurity' is a reasonable collaborative goal for the user to propose to this system. This check requires domain-specific knowledge and reasoners. Some other REQUEST acts, such as *Can you repeat what you said?,* would not be confirmed as a relevant shared domain goal and thus would not be interpreted as a PROPOSE for a shared goal. Rather, it would be interpreted using a model of the communication process and grounding.

In order to make the system as domain independent as possible, we balance this tension between the desire for domain-independent processing and the need for domain-specific processing by having the CPS manager generate hypotheses about the actual CPS act performed and then request the behavioral agent to evaluate their likelihood given the current problem-solving state. If the behavioral agent deems the hypothesis acceptable, then the CPS manager commits the act and thus changes the CPS model, thereby identifying what the system believes was the intended interpretation. If the behavioral agent finds a hypothesis unacceptable, the CPS manager can then suggest another possible interpretation and continue to do so until an acceptable one is found or until all hypothesized interpretations are exhausted. This evaluate-commit cycle (see Figure 4) is critical for enabling intention recognition that exploits strong linguistic context (i.e., the exact phrasing of utterances and the discourse context), as well as the strong context of

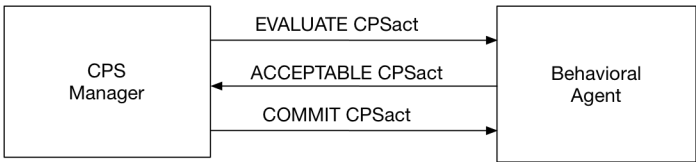

Figure 4. The EVALUATE-COMMIT cycle to determine whether a CPS act is acceptable before jointly committing to it.

the collaborative problem-solving state and domain reasoning provided by different back-end specialized reasoning engines.

Continuing with our running example, the system sends a set of messages to establish the joint goal to analyze food insecurity in Sudan:

CPS Manager → BA: (EVALUATE :content (ADOPT :id O1 :what C1 :as (GOAL)) :context …)
BA → CPS Manager: (ACCEPTABLE :content (ADOPT :id O1 :what C1 :as (GOAL)) :context …)
CPS Manager → BA: (COMMIT :content (ADOPT :id O1 :what C1 :as (GOAL)) :context …)

Here C1 is an identifier to the semantic representation of the action of analyzing food insecurity in Sudan, which is formally specified in the elided context arguments. In addition, the CPS manager issues a directive to the generation component to produce an acceptance of the proposal (the *OK* in utterance 2 in Table 1). Once the acceptance is generated, the system believes there is a shared joint goal with the user to analyze the food security situation in Sudan.

Looking at the formalism in a bit more detail, we see that a CPS act itself consists of two main parts: an operation and a collaborative problem-solving object. The operations model the interactions that the two agents take to establish joint intentions, including actions such as ADOPT and ABANDON, whereas the CPS objects are goals, constraints, situations, and the like. A brief summary of the key CPS acts is shown in Table 2. Each of these acts may take a number of arguments that provide further context for the operation. For instance, the ADOPT act might introduce a top-level goal or might introduce a goal that is a subgoal of an existing goal, e.g., a top level goal can be introduced with (ADOPT :id O1 :what C1 :as (GOAL)), whereas a subgoal could be (ADOPT :id O2 :what C2 :as (SUBGOAL :of O1)). More detail on the architecture of our CPS manager can be found in Galescu et al. (2018).

3.3 Defining Domain-Specific Capabilities

Before discussing the behavioral agent, we should address how the behavioral agent's domain-independent problem solving relates to the domain-specific reasoning and knowledge capabilities for the application domain. We will refer to these as the domain-specific reasoning engines (DSREs). At system startup time, each DSRE identifies its capabilities in terms of its input and output parameters indexed into a common ontology. The behavioral agent then uses this information when it needs information that matches one of the output parameters. Each parameter declaration includes, as appropriate, key information such as the required units for values. In addition, each input parameter is declared as being required (i.e., the values must be provided) or optional (i.e., generally this means the DSRE has default values if a value is not specified). In world modeling, many of the DSREs require or produce table-based data (e.g., the crop yields in different grid locations over ten years). To allow this, the specification language allow declaration of table formatting parameters.

Table 2. Key collaborative problem-solving acts.

Topic	ACT	Gloss	Example Sentence
Goals	ADOPT	Introducing a goal	Let's plan food aid for Sudan.
	SELECT	Focusing on existing goal	Let's return to the food problem.
	DEFER	Temporarily putting a goal aside	Let's work on this later.
	ABANDON	Abandoning a goal	I don't care about this anymore.
	RELEASE	Completing a goal	We're done.
Knowledge	ASSERTION	Making a claim	It's too hot to grow rice.
	ASK-IF	Asking a yes/no question	Is it too hot?
	ASK-WH	Asking a WH question	How cool does it need to be?
Reporting Problem-Solving Status	ACCEPTABLE	Goal is acceptable	OK. That's good.
	UNACCEPTABLE	Goal is unacceptable	That doesn't make sense.
	REJECTED	Goal is refused	No. I won't do that.
	FAILURE	Problem solving failed	I can't do that. I don't have enough wheat.
	ANSWER	Satisfying a question	We need cool nights.
	EXECUTION-STATUS	Reporting on progress in problem solving	I'm running the simulation. I found all the distribution areas.

As an example, Table 3 shows how the DSRE that encapsulates the pSIMS module identifies its capabilities, which includes computing crop production and other values for an area and timeframe, as described in Elliot et al. (2014). Looking at the declarations, we see this component is defined in terms of a set of input parameters and output parameters that specify what it can compute. All parameters are identified by an :ID-CODE and their data format is specified in the :FORMAT, along with a :UNIT field if it is a numerical quantity. For instance, the INPUT parameter LOCATION-FILE in Table 3 has the ID-CODE LOCATION and a FORMAT value of (RASTER "GTIff" 180 90).

Key to enabling the interoperability on different components is the declaration of parameters in terms of a common ontology. For this initial prototype, we hand-built the ontology drawing from a range of sources, including the ICASA ontology (White et al., 2013), the DSSAT parameter codes, and a few generic ontology types from the TRIPS ontology (e.g., LOCATION, TIME). The ID-CODE is drawn from this ontology and lets the system link the same parameters between different DSREs. From this declaration, we know this parameter identifies a LOCATION, and, furthermore, one that is specified as a gridded representation of the area of interest, (RASTER "GTIff" 180 90). The system also knows of other ways to represent locations, including ISO country codes, which are used by other DSREs. The planner uses a component called SPACEMAN that can map locations between different location formats as necessary; this also can map from natural language descriptions (e.g., country names) to various location formats.

As another example, the FEN_TOT input parameter encodes the amount of fertilizer used and has an ID-CODE drawn from the ICASA ontology. The pSIMS module requires the amount

Table 3. Excerpts from a declaration of a domain-specific reasoning engine.

```
(DEFINE-SERVICE :NAME PSIMS :COMPONENT CROPMODELLER
  :INPUT
    ( (INPUT :NAME LOCATION-FILE :GLOSS "image file of geographic region to simulate"
        :ID-CODE LOCATION :FORMAT (RASTER "GTiff" 180 90) :REQUIREMENTS :REQUIRED)
      (INPUT :NAME CRID :GLOSS "identifier of crop to simulate"
        :ID-CODE CRID :ID-CODE-CONSTRAINT (MAZ PML RIC SGG SBN WHB WHD WHT)
        :FORMAT ONT::CODE :REQUIREMENTS :REQUIRED)
      (INPUT :NAME PLYR :GLOSS "planting year(s) to simulate"
        :ID-CODE YEAR :ID-CODE-CONSTRAINT (RANGE 1980 2010)
        :FORMAT (OR ONT::NUMBER ONT::LIST) :REQUIREMENTS :REQUIRED)
      (INPUT :NAME FEN_TOT :GLOSS "total nitrogen in applied fertilizer"
        :ID-CODE FEN_TOT :ARGUMENTS (:LOCATION-FILE :CRID :PLYR)
        :UNIT (RATIO ONT::KG ONT::HECTARE) :FORMAT ONT::FUNCTION :REQUIREMENTS :OPTIONAL)
      …)
  :OUTPUT
    ( (OUTPUT :NAME PROD :GLOSS "global sum of harvest weight at maturity"
        :ID-CODE PROD :UNIT ONT::MEGATONNE :ARGUMENTS (:LOCATION-FILE :CRID :PLYR)
        :FORMAT (TABLE :PROD :LOCATION-FILE :CRID :PLYR)
      …)
```

to be specified in kg/hectare. Other modules may require fertilizer amounts to be expressed in different units; the planner uses a specialized component to convert between different unit types as desired. Note that FEN-TOT is actually a function dependent on the crop, location, and planting year. The declarations also specify whether the parameter must be specified in order to run the component (i.e., it is REQUIRED) or it is optional. FEN_TOT is optional and, if not specified, will be instantiated with a default level that the application retrieves based on the location. The CROP-ID, location, and planting year are all required.

Finally, a key part of the parameter declaration is the explicit representation of tabular data. Often a component will produce a table as its result. For example, the PROD parameter defined in the output declaration is a table that lists the production for each crop, location, and planting time specified in the inputs. Such tables must be explicitly defined so that the planner can extract information as necessary to prepare input to another component. For instance, once the production per crop per location per year is computed, the planner might sum over the location grid elements to produce a table that represents the total production for the whole area (e.g., South Sudan). In other cases, the planner might plan a series of calls to a component that iterates over the values in a table and then collects all the results to construct a new table for use as input to another DSRE. Given that DSREs are systems built by external parties with different uses in mind, the planner must be flexible enough to accommodate the data idiosyncrasies of each one. The careful declaration of the input and output parameters provides the information to accomplish this aim.

3.4 The Behavioral Agent

We can now describe the CWMS behavioral agent. In our previous applications, the behavioral agents have always been domain specific and built from the ground up for each domain. However, the behavioral agent of CWMS captures the generic problem-solving process itself,

Table 4. Some of the key collaborative problem-solving activities in world modeling.

Problem-Solving Task		Description	Examples of Goal Statements
1	Building causal models	Collaboratively build a causal model that captures key influences on a certain end condition or, alternatively, find a model that explains the relationship between two conditions (using prior knowledge, knowledge from reading, and user input)	*What are the key influences affecting food insecurity?* *How does a drought affect food availability?*
2	Planning and running simulation workflows	Given a causal model, collaboratively plan a workflow of existing simulation engines that could quantify the causal model (using knowledge of available simulation engines, and user input, especially with respect to generating approximations)	*What are the predictions for food insecurity in Sudan next year?* *How is migration affecting population at risk?*
3	Compare effects of varying input values (e.g., intervention analysis)	Given a causal model and/or a simulation workflow, collaboratively plan a series of runs under different conditions in order to explore how one condition affects another.	*How are crops affected if next year is an El Nino?* *What planting date would maximize the crop yield?*
4	Acquiring and preparing data sets	Given a set of information needs, collaboratively acquire or estimate this information using existing databases, acquiring data through reading existing documents, and developing approximations as necessary with the user.	*What are the soil characteristics in eastern South Sudan?* *How much maize was produced in each region of Sudan last year?*
5	Situation analysis	A very general behavior that typically invokes the other more specific behaviors. Given a problem statement to understand a situation, collaborate with user to facilitate their understanding of the actual conditions and possible interventions	*Let's analyze food insecurity in Sudan for the next two years*

abstracted away from specific domains. This distinction is similar to that found in general-purpose planning systems, where the planner itself is domain independent and the domain-specific aspects are found in the definitions of the operators. However, this correspondence only goes so far, as traditional planning is only a small part of overall problem-solving behavior. Other behaviors in CWMS include building causal models of phenomena, planning and evaluating interventions, and acquiring data from online sources and user interaction. Table 4 summarizes some different problem-solving tasks that are present in the world modeling domain. Note that any one dialogue will involve multiple problem-solving behaviors (e.g., to build a simulation workflow one might first need to build a plausible causal model).

There are two main functions that each collaborative problem-solving task must perform: intention recognition (i.e., given we are engaged in the current PS task, what does the user want to do now?) and behavior (i.e., once we know the new goal/subgoal in the task, how do we accomplish it?). The intention-recognition function supports the Evaluate/Commit protocol described above and the behavioral agent interacts with the CPS manager to arrive at an

interpretation of the latest user move (be it from language or a GUI action). The input to the intention recognition is a plausible CPS act computed by the CPS manager, which is then evaluated in the context of the current problem-solving task. The behavioral agent returns a response to the CPS manager that accepts, conditionally accepts, or rejects the interpretation. Once an interpretation is agreed upon and committed, the behavioral agent enters the behavior phase to perform the appropriate actions leading to a response.

Looking in more detail, each problem-solving task is represented as a state-transition network, namely a set of states where transitions between the states are labeled with a CPS act that signals the transition. Associated with each state is code that implements the system's "private" problem-solving behavior, which ultimately results in an attempt to establish the next shared CPS act that is captured in one of the outgoing transitions. In other words, the intention-recognition process involves either initiating a new problem-solving task or identifying the correct transition to take within the existing task. Once the transition is taken, the behavioral agent performs the behavior associated with the new state to further the problem-solving interaction. This behavior might involve planning, running simulations, acquiring data, or other reasoning, and ultimately terminates at the stage where user agreement would be necessary to continue (i.e., the system must initiate the next collaborative problem-solving act). The range of possible CPS acts will be captured by the set of outgoing transitions from the state. The new CPS act is proposed to the CPS manager and, if it is agreed to by the user, the transition is taken to the next problem-solving state in the task.

Note a key difference between the activity in the CPS manager and the activity of the behavioral agent: The CPS manager models the individual interactions between the system and user, while the behavioral agent models the problem-solving state in terms of committed (i.e., mutually agreed) problem-solving actions. In other words, the CPS manager models the dialogue in terms of proposals, acceptances, rejections, and so on. The behavioral agent, on the other hand, models the interaction in terms of the mutually agreed problem-solving actio ns (e.g., we are jointly analyzing food insecurity, we are agreeing to make a certain assumption, we are jointly building on a simulation model).

To make this clear, consider some of the interactions that occur in the example dialogue of Table 1. Specifically, let us start with the first six interactions:

(1) U: *Can we analyze food insecurity in Sudan next year.*
(2) S: *Should we look at child malnourishment rates?*
(3) U: *OK.*
(4) S: *Should I compute a baseline estimate based on available data?*
(5) U: *yes.*
(6) S: *OK. The percentage of malnourished children is about 39%.*

Figure 5 shows a path through the situation-analysis task that is invoked and executed during these six utterances. Note this is a single path through the model that captures this one dialogue. Other dialogues would take other paths in the model that are not shown.

As discussed earlier, statement (1) is transformed into a REQUEST to analyze food insecurity, which the CPS manager converts to a PROPOSAL of a new goal that the behavioral agent identifies as initiating the situation analysis task, where the condition of interest is food insecurity, the location of interest is Sudan, and the time of interest is next year, namely 2019. Once the COMMIT message is received by the behavioral agent, the first transition shown in Figure 5 is followed and the behavioral agent is in state SA-1. The behavior associated with this state is to identify how an abstract condition (i.e., food insecurity) might be analyzed in terms of

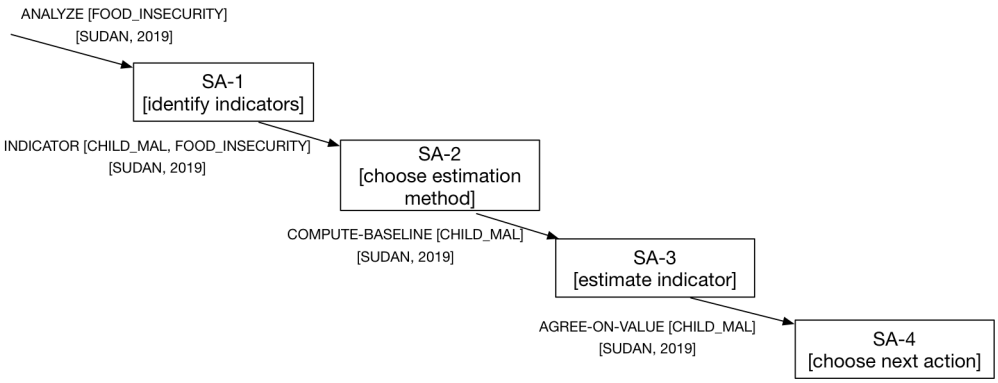

Figure 5. A trace through the situation analysis problem-solving model underlying the dialogue.

indicators that can be measured. Based on knowledge of a component from the IMPACT system, the system knows that the percentage of malnourished children is a possible indicator of food insecurity. The behavioral agent sends a proposal back to the CPS manager to reach agreement on using childhood malnutrition as an indicator (utterance 2). Once agreement is reached (utterance 3), the transition from SA-1 to SA-2 has been satisfied. Note this same model would account for a different dialogue in which the user says utterance (2) and the system makes utterance (3). Since the transitions are joint actions, it does not matter who actually made this initial proposal.

State SA-2 involves deciding on a method to estimate the indicator. There are a number of possibilities that a full system might consider. For instance, it might check if there is existing concrete data regarding the condition at the time and place (unlikely, as we are looking at the future) or it might search the web for documents that offer predictions and analyses for the condition at the time and place. The current prototype, however, is limited in its ability to search and retrieve such information from online sources. This is an area for future work. A third strategy, which we have implemented, is to see if the system has any reasoning engines that could compute an estimate for the indicator. In this case, CWMS does have such a component, namely a part of the IMPACT DSRE that can compute food insecurity measures. The system proposes to the user that it compute an estimate using baseline information (utterance 4). Once this is accepted (in utterance 5), the transition from SA-2 to SA-3 is made. In SA-3, the system invokes the planner to estimate the percentage of child malnourishment in Sudan in 2019. In this case, the plan is trivial, as IMPACT contains all the default data needed to compute the estimated value. Once this plan is executed (i.e., the IMPACT module has been invoked and returns a result), the behavioral agent then sends the CPS manager a request to INFORM the user of the result (utterance 6) as a way to establish the transition at the bottom of Figure 5.

The next segment of the dialogue involves more complex behavior:

(7) U: *What if 2019 is an El Nino year?*
(8) S: *I can estimate the effect with the following plan. Does that look good?*
(9) U: *Yes.*
(10) S: *OK. The percentage of malnourished children increases by 5%.*

The intention recognition for utterance (7) occurs in the context of the AGREE-ON-VALUE transition from state SA-3 in Figure 5. The CPS manager is expecting an acceptance or rejection of utterance (6), but utterance (7) appears to be neither. This is commonplace in human dialogue, in which a proposal can be implicitly accepted if the response can be interpreted as continuing the

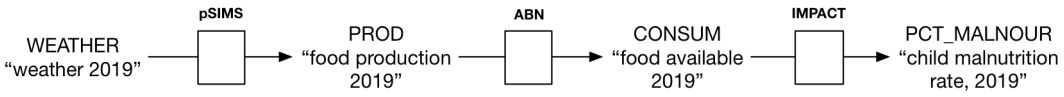

Figure 6. An initial causal graph connecting weather to food insecurity.

current problem-solving process. The system does this by attempting intention recognition on utterance (7) as though utterance (6) had been accepted (assuming the response was not an explicit rejection!). Utterance (7) is recognized as a proposal to compute an alternative analysis assuming that 2019 is an El Niño year, an expected continuation of the problem-solving process. The behavioral agent receives an acceptance of the value for childhood malnutrition from the CPS manager, which completes the transition to SA-4 at the bottom of Figure 5.

Following this, the CPS manager sends the behavioral agent the (already derived) interpretation of utterance (7) and the behavioral agent in response initiates a subtask to compute and compare a set of results, with the control parameter being whether 2019 is an El Niño year. The first step of this subtask is to initiate a further subtask to identify a causal model that relates the target condition (child malnutrition) with the control condition (2019 as El Niño year or not). The CWMS system attempts to construct such a model using its knowledge about available domain-specific reasoning engines, as finding a chain through the input and output parameters of these models would suggest a method for computing the values. Figure 6 shows an initial chain linking the two parameters.

Although this identifies a possible causal chain through a suite of known reasoning engines, there are problems that must be resolved before this becomes an executable workflow. First, each of the DSREs have other input parameters that are required – some values can be identified in the immediate context (e.g., LOCATION is Sudan, year is 2019), others might need to be retrieved or looked up (e.g., the main crops grown in Sudan), and others might need approximations developed in collaboration with the user. A key area where approximations must be used is weather. The agricultural models used in pSIMS and DSSAT need detailed day-to-day weather data to simulate the growth of the crops. However, we clearly lack such a model of the weather in 2019, as it has not happened yet! In fact, for pSIMS we only have weather records for the years 1980 to 2010. To deal with this problem, we developed a weather estimation operator in CWMS that runs desired configurations on past weather data, then uses an interpolation function to approximate the values for a future date. As a default interpolation, when we have no insight into the future weather, we use a simple weighted average that gives slightly more relevance to more recent years. If we know something about the future weather, say that it is an El Niño year, then we can use an interpolation function over past years in which previous El Niño years are given more weight. Using this capability, CWMS revises its plan in Figure 6 to add the segment approximating yields in 2019, as shown in Figure 7.

Once the system thinks it has generated a valid plan, it presents the plan to the user and asks for comments and/or approval (utterance 9). At this stage, the user might suggest modifications and revisions to the plan, completely reject it, or accept it. In our example, the user accepts the plan (utterance 10), which the system then executes under the two control conditions (unspecified default weather vs. El Niño weather). Plan execution for the most part involves invoking the DSREs in the appropriate order, after making necessary data conversions to match the required data formats for each DSRE. Once CWMS completes the execution, it summarizes the effect of the new assumption in terms of the change in rate of malnourished children (utterance 10). Note that utterance (11) involves yet another implicit acceptance of the result. Once this is recognized,

Figure 7. Revised plan adding weather estimation to predict future production.

the subtask of comparing effects is completed and the behavioral agent is once again back in state SA-4, ready to interpret the user's new request, which is how to mitigate the effects.

4. Concluding Remarks

In this article, we have discussed a framework that can manage dialogues for substantially more complex tasks than possible with previous approaches. This is demonstrated in a detailed discussion of the CWMS system for the domain of world modeling. Furthermore, the system applies generally to domains that can be cast as collaborative problem solving. We have shown how the dialogue management can be integrated with domain-dependent reasoning engines based solely on the declaration of their reasoning capabilities in an agent-based framework. CWMS can engage in complex collaborative problem solving as required for modeling world processes. We note that the current implementation is only a prototype and much work remains to be done to produce a robust, fully functional system that can be used by analysts.

One of the remaining key challenges is that, unlike the example dialogue we used above that was language dominant, complex world modeling will often require significant use of GUIs for communication. The system must be able to present information and summarize results using maps, charts, and other graphical devices. Furthermore, these GUI components, like DSREs, will generally be developed by other organizations. We plan to integrate these GUI engines in a similar way to DSREs, where each GUI declares its capabilities at system start up. The additional complication is that the dialogue system must be constantly updated on the information currently being presented and the options (e.g., menu items, gestures) that are available. In essence, the active GUIs provide the context for the dialogue, and many system actions may need to be accomplished via the GUI's API.

While the challenges remaining are great, we are optimistic that the partition of responsibilities we have outlined here and the domain-independent framework for collaborative problem solving will provide the environment for building truly useful systems.

Acknowledgements

We thank the reviewers for excellent comments and suggestions for improving the paper. This research was supported by the DARPA (ARO contracts W911NF-14-0391, W911NF-17-1-0047 and W911NF-18-1-0464).

References

Allen, J. F. (1995). *Natural language understanding*. Redwood City, CA: Benjamin Cummings.

Allen, J., Bahkshandeh, O., de Beaumont, W., Galescu, L. & Teng, C. M. (2018). Effective broad-coverage deep parsing. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (pp. 4776–4783). New Orleans, LA: AAAI Press.

Allen, J., Blaylock, N., & Ferguson, G. (2002). A problem solving model for collaborative agents. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2* (pp. 774–781). Bologna, Italy: ACM.

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., & A. Stent. (2000). An architecture for a generic dialogue shell. *Natural Language Engineering*, *6*, 213–228.

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W. (2007). PLOW: a collaborative task learning agent. *Proceedings of the Twenty-Second AAAI Conference on Artificial intelligence* (pp. 1514–1519). Vancouver, BC: AAAI Press.

Allen, J. F. & Perrault., C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence, 15*, 143–178.

Allen, J. F. & Teng, C. M. (2017). Broad coverage, domain-generic deep semantic parsing. *Proceedings of the AAAI Spring Symposium on Computational Construction Grammar and Natural Language Understanding* (pp. 108–115). AI Access Foundation.

Baker, C. F., Fillmore, C. J, & Lowe, J. B. (1998). The Berkeley FrameNet Project. *Proceedings of the Seventeenth International Conference on Computational Linguistics* (pp. 86–90). Montreal, Canada: ACL.

Blaylock, N., & Allen, J. (2005). A collaborative problem-solving model of dialogue. *Proceedings of the Sixth SIGdial Workshop on Discourse and Dialogue* (pp. 200–211). Lisbon, Portugal: ISCA.

Bohus, D., & Rudnicky, A. I. (2009). The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, *23*, 332–361.

Cohen, P. R., & Levesque, H. (1990). Rational interaction as a basis for communication. In P. R. Cohen, J. Morgan, & M. E. Pollack (Eds.), *Intentions in communication*. Cambridge, MA: MIT Press.

Cooper, R. (1997). Information states, attitudes and dialogue. *Proceedings of the Second Tbilisi Symposium on Language, Logic and Computation* (pp. 15–20). Tbilisi State University.

Elliott, J., Kelly, D., Chryssanthacopoulos, J., Glotter, M., Jhunjhnuwala, K., Best, N., Wilde, M. & Foster, I. (2014). The parallel system for integrating impact models and sectors (pSIMS). *Environmental Modelling & Software*, *62*, 509–516.

Fellbaum, C. (Ed.) (1998) *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.

Ferguson, G. & Allen, J. (2007). Mixed-initiative systems for collaborative problem solving. *AI Magazine*, *28*, 23–32.

Gabaldon, A., Langley, P., & Meadows, B. (2014). Integrating meta-level and domain-level knowledge for task-oriented dialogue. *Advances in Cognitive Systems, 3,* 201–219.

Galescu, L., Teng, C. M., Allen, J. & Perera, I. (2018). COGENT: A generic dialogue system shell based on a collaborative problem solving model. *Proceedings of the Nineteenth Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 400–409). Melbourne, Australia: ACL.

Grosz, B. J. & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86*, 269–357.

Gyori, B. M., Bachman, J. A., Subramanian, K., Muhlich, J. L., Galescu, L., & Sorger, P. K. (2017). From word models to executable models of signaling networks using automated assembly. *Molecular Systems Biology*, *13*, 954.

Hatfield-Dodds, S., Adams, P. D., Brinsmead, T. S., Bryan, B. A., Chiew, F. H. S., Finnigan, J. J., Graham, P. W., Grundy, M. J., Harwood, T., McCallum, R., McKellar, L. E., Newth, D., Nolan, M., Schandl, H., & Wonhas, A. (2015). *Australian National Outlook 2015: Economic activity, resource use, environmental performance and living standards, 1970–2050*. Canberra, Australia: CSIRO.

Jones, J. W., Hoogenboom, G., Porter, C. H., Boote, K. J., Batchelor, W. D., Hunt, L. A., Wilkens, P. W., Singh, U., Gijsman, A. J., & Ritchie, J. T. (2003). DSSAT cropping system model. *European Journal of Agronomy, 18,* 235–265.

Kim, S., Salter, D., DeLuccia, L., Son, K., Amer, M. R., & Tamrakar, A. (2018). SMILEE: Symmetric multi-modal interactions with language-gesture enabled (AI) embodiment. *Proceedings of the Sixteenth Annual Conference of the North American Chapter of the ACL: Human Language Technologies* (pp. 86–90). New Orleans, LA: ACL.

Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation Journal*, *42*, 21–40.

Larsson, S., & Traum, D. R. (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, *6*, 323–340.

Lison, P., & Kennington, C. (2016). OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules. *Proceedings of the Fifty-Fourth Annual Meeting of the ACL: System Demonstrations* (pp. 67–72). Baltimore, MD: ACL.

Litman, D. J., & Allen, J. F. (1987). A plan recognition model for subdialogues in conversations. *Cognitive Science*, *11*, 163–200.

Ljunglöf, Peter. 2009. trindikit.py: An open-source Python library for developing ISU-based dialogue systems. *Proceedings of the First International Workshop on Spoken Dialogue Systems Technology*. Kloster Irsee, Germany.

Lochbaum, K. E., Grosz, B. J., & Sidner, C. L. (1990). Models of plans to support communication: An initial report. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 485–490). Boston, MA: AAAI Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. *Proceedings of the Fifty-Second Annual Meeting of the ACL: System Demonstrations* (pp. 55–60). Baltimore, MD: ACL.

Manshadi, M. H., Allen, J., and Swift, M. (2008). Toward a universal underspecified semantic representation. *Proceedings of the Thirteenth Conference on Formal Grammar*. Hamburg, Germany.

Marchand, P., Carr, J. A., Dell'Angelo, J., Fader, M., Gephart, J. A., Kummu, M., Magliocca, N. R., Porkka, M., Puma, M. J., Ratajczak, Z., & Rulli, M. C. (2016). Reserves and trade jointly determine exposure to food supply shocks. *Environmental Research Letters*, *11*, 095009.

Perera, I., Allen, J., Galescu, L., Teng, C. M., Burstein, M., McDonald, D., Rye J., & Friedman, S. (2017). Natural language dialogue for building and learning models and structures. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 5103–5104). San Francisco, CA: AAAI Press.

Perera, I., Allen, J. F., Galescu, L., & Teng, C. M. (2018). A multimodal dialogue system for learning structural concepts in blocks world. *Proceedings of the Nineteenth Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 89–98). Melbourne, Australia: ACL.

Quick, D., & Morrison, C. T. (2017). Composition by conversation. *Proceedings of the Forty-Third International Computer Music Conference* (pp. 52–57). Shanghai, China.

Rich, C., & Sidner C. L. (2012). Using collaborative discourse theory to partially automate dialogue tree authoring. In Y. Nakano, M. Neff, A. Paiva, & M. Walker (Eds.), *Intelligent virtual agents*. Berlin, Heidelberg: Springer.

Robinson, S., Mason-D'Croz, D., Sulser, T., Islam, S., Robertson, R., Zhu, T., Gueneau, A., Pitois, G., & Rosegrant, M. W. (2015). *The international model for policy analysis of agricultural commodities and trade (IMPACT): Model description, version 3.* IFPRI Discussion Paper 1483. Washington, DC: IFPRI.

Schewe, J., Otto, C., & Frieler, K. (2017). The role of storage dynamics in annual wheat prices. *Environmental Research Letters*, *12*, 054005.

White, J. W., Hunt, L. A., Boote, K. J., Jones, J. W., Koo, J., Kim, S., Porter, C. H., Wilkens, P. W., & Hoogenboom, G. (2013). Integrated description of agricultural field experiments and production: The ICASA Version 2.0 data standards. *Computers and Electronics in Agriculture*, *96*, 1–12.

Williams, J., Raux, A., & Henderson, M. (2016). The dialog state tracking challenge series: A review. *Dialogue & Discourse*, *7*, 4–33.

Appendix

For further details on CWMS and related TRIPS-based systems, including available open source code, the reader is referred to these links:

- `www.cs.rochester.edu/research/trips/lexicon/browse-ont-lex.html` for browsing the TRIPS lexicon and ontology;
- `trips.ihmc.us/parser` for on-line interfaces to the TRIPS parser customized for different domains, including CWMS;
- `github.com/wdebeaum/cogent` for source code for the generic dialogue shell based on collaborative problem solving (Cogent), which includes the parser, dialogue management, and the CPS manager, but which does not yet include the behavioral agent described in this paper.