Moving QA Towards Reading Comprehension Using Context and Default Reasoning

Christine Clark, Daniel Hodges, Jens Stephan, Dan Moldovan
Language Computer Corporation
Richardson, Texas, 75080
{christine,daniel,jens,moldovan}@languagecomputer.com

Abstract

For a question answering system to understand the underlying assumptions and contextual features of natural language, a contextually and semantically sensitive knowledge representation and reasoning module is essential. This paper proposes a three-layered knowledge representation coupled with reasoning mechanisms for defaults and contexts as a solution for pushing question answering towards a state capable of basic reading comprehension. A travel scenario is presented as a vehicle for demonstrating the complexities of this problem, and the ways in which semantic relations, contexts (for conditionality, planning, time, and space), and defaults address these complexities.

Introduction

Natural language is rich in implicit and explicit contexts in addition to default assumptions that humans intuitively capture in their mental process. For a machine such as a question answering system to perform the same task, a careful and precise knowledge encoding scheme is required, as well as accompanying reasoning mechanisms for contexts and defaults. Consider the following travel scenario provided by AQUAINT:

1. The Amtrak train arrived at Union Station Washington D.C. from New York City at 10 am on March 15, 2005.
2. It was an Acela Express, number 176, scheduled to depart 30 minutes later on its return trip to the Big Apple.
3. John took a taxi from his home in Georgetown and arrived at the station 15 minutes before 176's scheduled departure.
4. He planned to meet his financial adviser, Bill, at Citibank, the same day at 2 pm.
5. If John's train ran on time, the three-hour trip will get him there on time.
6. If not, John intended to call Bill and push back the appointment two hours.

After a quick glance, a human can easily comprehend the locative, temporal, and conditional aspects of the text, and can answer questions like "What time did John arrive at Union Station?" or "If the train did not run on time after departing from Union Station, what would happen?". Currently no automated question answering system can reliably handle this "simple" reading comprehension exercise.

Recognizing, representing, and reasoning over contexts, both explicit and implicit, is a major roadblock in achieving this task. Without proper context representation, it is not possible to accurately answer questions that require lifting (McCarthy 1993), like "Did John call Bill?", since this event would only take place if the train is indeed late and John fulfills his intentions to do so. Furthermore, multiple instances of the same event, e.g. the arrival of the train, can be correctly identified by its context, which allows the engine to focus on "the task at hand" (Lenat 1998).

LCC demonstrated the potential of context detection and reasoning by integrating a temporal context module into a state of the art QA system (Moldovan et al 2005). However, for the field of QA to move towards the level of basic human reading comprehension, a richer set of both physical (e.g. spatial, temporal) and possible world contexts (e.g. subjectivity, planning, conditionality) as well as methods for default reasoning need to be incorporated.

As motivated by the above mentioned scenario, we have developed a model for representing and reasoning within contexts that enables some contextual inferences and defaults. This paper will outline the details of a first order logic based representation, as well as the knowledge and inference mechanisms required to reason with this representation.

Approach

To achieve a question answering system that is capable of understanding and reasoning with the semantics and contexts embedded in natural language exemplified by the travel scenario, we propose layering conditional, planning, temporal, and spatial aspects onto our existing first order logic based knowledge representation. The details of this "three-layered" approach is detailed in the section "A Semantically and Contextually Rich Knowledge Representation".

To fully exercise the power of this enriched knowledge

representation, a hierarchy of trigger rules and assumption axioms should be added to enable contexts and propagate context information across the discourse of the scenario. This is instrumental for questions with contextual constraints and is demonstrated in the section "Answering Contextually Constrained Questions". Further, to answer open ended questions and to reason with defaults, we propose the addition of mechanisms in the inference engine to track new inferences and reliably enable defaults with a special purpose reasoner. The details of this approach are outlined in the section "Answering Hypothetical Questions".

A Semantically and Contextually Rich Knowledge Representation

To represent the travel scenario accurately in first order logic, it is necessary to encode the syntactic and semantic layers of the sentences, and use these as signals for marking context boundaries. Unless otherwise indicated, all formulas are universally quantified, variable names are preceded by a question mark, and all other predicate arguments are skolemized constants. We will be using the following sentence to demonstrate our representation:

He (John) planned to meet his financial adviser, Bill, at Citibank the same day (March 15, 2005) at 2pm.

In the first layer of the representation, the first order logic structure is derived from the syntactic parse of the text and each token in the text is converted to a predicate of the structure *lemma_part-of-speech* (Rus and Moldovan 2002).

John_NN(x27) & plan_VB(e6,x27,x51) & to_TO(e6,e7) & meet_VB(e7,x27,x34) & his_PRP$(x34,x27) & financial_NN(x32) & adviser_NN(x33) & nn_NNC(x34,x32,x33) & Bill_NN(x50) & at_IN(e7,x35) & Citibank_NN(x35) & same_JJ(v10,x36) & day_NN(x36) & at_IN(e7,x39) & 2_NN(x37) & pm_NN(x38) & nn_NNC(x39,x37,x38)

The second layer encodes the semantic relations found in the text by introducing a predicate labeled with the type of semantic relation whose arguments are the heads of the phrases that participate in the relation. The **SYN** relation states that Bill is synonymous with financial adviser. The **TMP** and **LOC** predicates relate the time and location to the meet event.

SYN_SR(x50,x34) & LOC_SR(x35,e7) & TMP_SR(x36,e7) & TMP_SR(x39,e7)

The third layer marks the context boundaries of the sentence using predicates that map to the type of context. The types of context that will be required to represent this scenario are temporal, spatial, conditional, planning, and assumed contexts.

Temporal and Spatial Contexts

Temporal contexts are represented by introducing SUMO[1] (Niles and Pease 2001) predicates for the Allen primitives (Allen 1991) into the third layer of the representation. A thorough explanation of the temporal context representation is provided in (Moldovan et al 2005). When time related predicates are found in the first layer of the logic form they are removed and replaced by appropriate temporal context predicates in the third layer. Here the **time_CTMP** predicate defines the time of the context and the **during_CTXT** predicate attaches that time to the meet event to indicate that the meeting is to occur in the temporal context of 2 PM, March 15, 2005.

during_CTXT(e7,t10) &
time_CTMP(2005,03,15,14,0,0)

We represent spatial contexts through predicates that indicate the nature of the relation, such as destination, source, and location. The **at_CTXT** predicate is used in this example to indicate that Citibank is the locative context of the meeting.

at_CTXT(e7,x35)

Conditional and Planning Contexts

Conditional contexts are used to express events and objects that only occur in some hypothetical world. These hypothetical worlds are defined by the preconditions of the contexts. Similarly, planning contexts capture events and objects that exist in hypothetical worlds defined by intentions, plans, and wills. To represent these contexts, predicates labeled with the type of context *(planning or conditional)* are introduced with a context constant to track the context, and the remaining arguments are the identifying variables for the events scoped by the context. An implication is formed so as to prevent the knowledge base from asserting the contents of the contexts without a trigger rule. In the example, the detection of the signal words "planned to" informs us that the assertion, "meet his financial adviser...", occurs in a planning context. The signal words are removed from the first layer of the logic form and the assertion becomes implied by a **planning_CTXT** predicate.

planning_CTXT(p2,e7) → meet_VB(e7,x27,x34) & at_IN(e7,x35) & Citibank_NN(x35) & same_JJ(v10,x36) & day_NN(x36) & at_IN(e7,x39) & 2_NN(x37) & pm_NN(x38) & nn_NNC(x39,x37,x38) & _time_NE(x39) & time_CTMP(2005,03,15,14,0,0) & LOC_SR(x35,e7) & TMP_SR(x36,e7) & TMP_SR(x39,e7) & at_CTXT(e7,x35) & during_CTXT(e7,t10)

For both planning and conditional contexts we include a trigger rule that allows the prover to enter the context if the preconditions of the context are met. Conditional contexts are simply triggered by the preconditions of their context. Planning contexts are triggered when there is evidence in the knowledge base that the plan was fulfilled. Thus, the contents of the planning context are the triggers for the planning context. So if John plans X, and later we find he executed X, then that planning context is enabled.

meet_VB(e7,x27,x34) → planning_CTXT(p2,e7)

[1] Suggested Upper Merged Ontology

Assumed Contexts

Assumed contexts are an important part of our default reasoning implementation. The **assume_CTXT** predicate indicates to the prover that it is to be assumed that the preconditions of a context have been met, unless there is evidence to the contrary. The single argument of the predicate references the context for which preconditions are to be assumed. Each context that is enabled with default reasoning should have an associated trigger rule where the antecedent is the **assume_CTXT** predicate and the consequent is the appropriate context predicate. In this example, we want to be able to reason by default that John's meeting occurs as planned. This is accomplished by making **assume_CTXT** imply **planning_CTXT**.

assume_CTXT(p2) → planning_CTXT(p2,e7)

Answering Contextually Constrained Questions

The context resolution module employs our first order logic resolution style theorem prover, COGEX, that has been adapted for natural language text processing (Moldovan et al 2003). The Set of Support (SOS) search strategy is employed by the prover to partition the axioms into those that have support and those that are considered auxiliary (Wos 1988). This strategy restricts the search such that a new clause is inferred if and only if one of its parent clauses come from the SOS. The inference rule sets are based on hyperresolution, paramodulation, and demodulation. Hyperresolution is an inference rule that performs multiple steps at once. Paramodulation introduces the notion of equality substitution. Hyperresolution and paramodulation inference rules allow for a more compact and efficient proof by condensing multiple steps into one. Below we present solutions to questions posed in the travel scenario that require temporal, spatial, conditional, and planning contexts. Take the following:

[s1] *"The Amtrak train arrived at Union Station Washington D.C. from New York City at 10 am on March 15, 2005."*
[s2] *"It was an Acela Express, number 176, scheduled to depart 30 minutes later on its return trip to the Big Apple."*
[s3] *"John took a taxi from his home in Georgetown and arrived at the station 15 minutes before 176's scheduled departure."*
[s5] *"If John's train ran on time, the three-hour trip will get him there on time."*
[q1] *"What time did John arrive at Union Station?"*
[q2] *"If it departed on time, and the train ran on schedule, what time would the train arrive in New York?"*

The goal for these two questions is to identify and resolve the temporal contexts connected to the respective arrival events in question. A human would answer these questions with "10:15 am" and "1:30 pm" without much trouble. An automated inference engine however requires a series of steps to arrive at these conclusions, which include:

(1) Detecting the implicit arrival event of the train in New York
(2) Disambiguating the correct arrival event by applying the spatial contextual constrains "at Union Station" and "in New York".
(3) Propagating the temporal contexts from the absolute time arrival event to the relative events of the question.

We propose an architecture that detects the contextual constraints of a question, resolves context in candidate answers, and propagates context information among the answer passages.

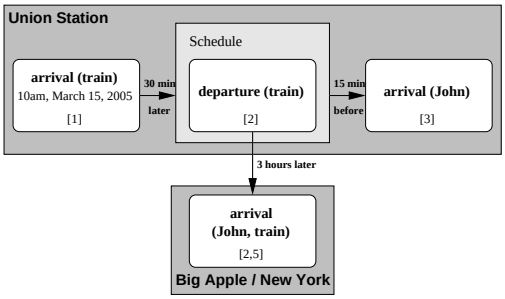

Figure 1: Constraints: temporal, spatial, planning

Figure 1 illustrates the context relations that need to be resolved to enable time stamp propagation to answer the questions. The dark boxes represent the spatial context of the events and the white boxes represent the temporal context of the events. Since there are several arrival events of the same agents, the spatial contexts are needed to disambiguate. The train's arrival event at Union Station is the only event anchored in absolute time and all other events are relative to it. The arrows labeled with time values show the details of this offset. The light grey box represents the planning context introduced in the scenario by the scheduling event. The implicit arrival in the Big Apple (New York City) is relative to the actual departure event at Union Station, while the other events are chained to the scheduling context.

To answer [q1], the correct arrival event for John has to be chosen based on the spatial constraint "Union Station". The time stamp for this event can now be propagated using the absolute time of the train's arrival at Union Station and applying the relative operators "30 minutes later" and "15 minutes before" on the propagation chain.

The following is the final proof output of the logic prover for [q1]. The proof begins by making the necessary assertions from the scenario:

Scenario Assertions
(1) John_NN(x27) & arrive_VB(e5,x27,x31) & station_NN(x6) & LOC_SR(x7,e5) & destination_CTXT(x6,e5) & during_CTXT(e5,t12) & time_CTMP(t12,2005,03,15,10,15,0)

The prover utilizes the following synonymy axiom to equate "Union Station" with "station":

Synonymy axiom
(2) -station_NN(x6) | (Union_NN(x5) & Station_NN(x6) & nn_NNC(x7,x5,x6))

(3) Union_NN(x5)
(4) Station_NN(x6)
(5) nn_NNC(x7,x5,x6)

The prover inserts the question logic form into the search space and unifies all of the negated predicates in the logic form with predicates in the knowledge base to produce a refutation:

[q1] Logic Form
(6) -John_NN(?x3) | -arrive_VB(?x1,?x3,?x4) | -Union_NN(?x5) | -Station_NN(?x6) | -nn_NNC(?x7,?x5,?x6) | -LOC_SR(?x7,?x1) | -destination_CTXT(?x7,?x1) | -during_CTXT(?x1,?x2) | -time_CTMP(?x2,?x8,?x9,?x10,?x11,?x12,?x13)

The time of John's arrival is extracted from the **time_CTMP** predicate in the answer logic form that the question unified with. The prover guarantees the time is correct by making sure that the event that the time is associated with occurs in the spatial context of Union Station:

John arrived at Union Station at 10:15 AM on March 15, 2005.

To resolve [q2], we need to detect the implicit arrival event at the Big Apple for John and the train. This can be achieved by common sense knowledge axioms. At first, the time stamp propagation chain is broken inside the scheduling event. However, the scheduling context is qualified by the conditional context of the question by stating "if it departed on time, and the train ran on schedule". We can therefore construct a time stamp propagation chain to the absolute time event by linking to the time stamp of the scheduling context.

Here we present the proof for [q2]. The prover first asserts the conditional contexts in the question:

Assertion
(1) train_NN(x3) & depart_VB(e3,x3,d0) & on_time_RB(v5,e3) & run_VB(e8,x3,x40) & on_IN(e8,x5) & schedule_NN(x5)

The assertion fulfills the preconditions of the planning context, that require us to know if the train departed on schedule. Axioms (2) and (3) below equate "on time" with "on schedule":

World Knowledge Axioms
(2) -on_time_RB(?x1,?x2) | (on_IN(?x2,?x3) & schedule_NN(?x3))
(3) -on_IN(?x2,?x3) | -schedule_NN(?x3) | on_time_RB(?x1,?x2)
(4) schedule_NN(?x1) & on_IN(e3,?x1)
(5) on_time_RB(?x1,e8)

Once we know that the train departed on schedule, we can enter the planning context via the trigger rule in (6). From the planning context we infer that the train's destination is the "Big Apple":

Planning Context
(6) -on_IN(e3,s1) | -schedule_NN(s1) | planning_CTXT(p1,e3) -planning_CTXT(p1,e3) | (to_TO(x4,x11) & Big_Apple_NN(x11) & destination_CTXT(x4,x11))
(7) planning_CTXT(p1,e3)
(8) to_TO(x4,x11) & Big_Apple_NN(x11) & destination_CTXT(x4,x11)

Now, we use axioms (9) and (11) below to equate "Big Apple" with "New York City" and "John's train" with "train":

Synonymy Axioms
(9) -Big_Apple_NN(x11) | New_York_City_NN(x11)
(10) New_York_City_NN(x11)
(11) -train_NN(x3) | (John_NN(x27) & _s_POS(x3,x27))
(12) John_NN(x27) & _s_POS(x3,x27)

The question's assertion also informs us that the train ran on schedule. Using this knowledge and trigger rule (13) below, we infer that an on time train should enable the conditional context **c1**. This context informs us of the duration of the trip:

Conditional Context
(13) -John_NN(x27) | -_s_POS(x3,x27) | -train_NN(x3) | -run_VB(e8,x3,x40) | -on_time_RB(v3,e8) | conditional_CTXT(c1,e8)
(14) -conditional_CTXT(c1,e8) | (trip_NN(x4) & Time_Length(t3,3,0,0) & duration_CTXT(x4,t3))
(15) conditional_CTXT(c1,e8)
(16) duration_CTXT(x4,t3) & Time_Length(t3,3,0,0) & trip_NN(x4)

The axiom below tells us that an arrival event occurs when there is a departure and a trip. This axiom also calculates the time of that arrival by adding the duration of the trip to its departure time:

Arrival Axiom
(17) -depart_VB(e3,x3,d0) | -trip_NN(x4) | -duration_CTXT(x4,t3) | -Time_Length(t3,3,0,0) | (arrive_VB(e13,x3,d2) & during_CTXT(e13,t4) & time_CTMP(t4,2005,3,15,13,30,0) & TMP_SR(t4,e13))
(18) during_CTXT(e13,t4) & TMP_SR(t4,e13) & arrive_VB(e13,x3,d2) & time_CTMP(t4,2005,3,15,13,30,0)

The destination axiom below declares that an object that departs on a trip with a destination will arrive at that destination:

Destination Axiom
(19) -depart_VB(?x1,?x2,?x3) | -trip_NN(?x4) | -to_TO(?x4,?x5) | -destination_CTXT(?x4,?x5) | -arrive_VB(?x6,?x2,?x7) | (LOC_SR(?x5,?x6) & at_CTXT(?x2,?x5))
(20) at_CTXT(x3,x11) & LOC_SR(x11,e13)

The question logic form is inserted into the search and a refutation is found:

Question
(21) -train_NN(?x1) | -arrive_VB(?x2,?x1,?x3)

| -New_York_City_NN(?x4) | -TMP_SR(?x5,?x2) | -LOC_SR(?x4,?x2) | -during_CTXT(?x2,?x5) | -at_CTXT(?x1,?x4) | -time_CTMP(?x5,?x6,?x7,?x8,?x9,?x10,?x11)

The arrival time in the question will unify with the time calculated by the arrival axiom. We verify that this is the correct arrival time by making sure that the train's arrival occurs in the spatial context "New York":

The train will arrive in New York at 1:30 PM on March 15, 2005.

Answering Hypothetical Questions

In the previous examples we used a proof-by-contradiction method to find an answer. However, in the following scenario we are confronted with open-ended questions which make proof-by-contradiction inappropriate:

[s5] *If John's train ran on time, the three-hour trip will get him there on time.*
[s6] *If not, John intended to call Bill and push back the appointment two hours.*
[q9] *If the train departs on time and runs on schedule, what will happen?*
[q5] *If John's train did not run on time after departing from Union Station, what would happen?*

The new goal is to discover what will happen in the hypothetical world where *John's train runs on time or doesn't run on time*. The answer to [q9] is clearly that John will call Bill and push back the appointment two hours, it is easy for a human to arrive at this conclusion. The steps for an inference engine are more complex and include:

(1) Deriving "John intended to call Bill and push back the appointment two hours" from the hypothetical condition: "John's train did not run on time".
(2) Entering the planning context of John's intentions via an assumed context to derive: "John calls Bill and pushes back the appointment two hours"

We propose a solution to this problem that requires tracking the side effects of adding hypothetical assertions to a knowledge base as well as applying default reasoning to explore answer paths for which sufficient trigger rules are not available. This lifts conclusions out of their context that should be returned as qualified answers to the user.

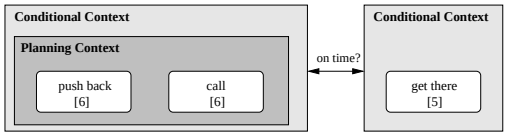

Figure 2: Conditional and Planning Contexts

Figure 2 illustrates the hypothetical propositions asserted by the scenario. The events of the conditional contexts are surrounded by a bounding box to illustrate that this knowledge is only true in a world where the assumptions of its context are true. The events of the planning context are surrounded by a box which is inside the conditional context's box to illustrate that this knowledge is only true in a world where the assumptions of both of their surrounding contexts are true.

In the above example, the conditional contexts embedded in the scenario text are represented as described in section "A Semantically and Contextually Rich Knowledge Representation", and so the statements, "the three-hour trip will get him there on time" and "John intended to call Bill and push back the appointment two hours" are preconditioned on whether the train runs on time or not. In sentence [s6], John's intentions must be represented inside a planning context. The scenario tells us nothing of what John will do, only what he plans on doing. Just as with the conditional context, if we do not explicitly scope John's intentions inside a planning context, his intentions will be treated as unqualified facts in the knowledge base. Further, this planning context occurs inside a conditional context. Until we know for sure that the train does not run on time, we can say nothing about John's plans. Thus, it is necessary to nest the contexts, which is easily supported by the knowledge representation.

The questions are preconditioned on whether the train will run on time, but the system interprets these preconditions as hypothetical assertions that should trigger conditional contexts in the knowledge base. The hypothetical clause in the questions define possible worlds, and the set of propositions that are inferred from the scenario knowledge base together with the hypothetical assertions from the question, is the set of events that are true in this possible world. We can get this information by exploring the search trace for the inferred clauses derived from the hypothetical.

To handle entering John's planning context we need default reasoning. This is achieved by adding **assume_CTXT** predicates to the knowledge base that do not contradict the currently inferred knowledge. The assume context assertions lift constrained facts out of their context and allow the prover to reason within that context. The mechanism for default reasoning integrated with COGEX is illustrated in Figure 3.

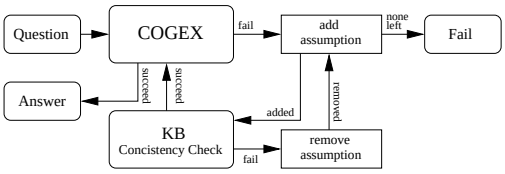

Figure 3: Reasoning with Defaults

Before **assume_CTXT** predicates are added to the knowledge base, COGEX attempts to find new inferences derived from the question axiom. If one cannot be found, the default reasoning module incrementally adds **assume_CTXT** predicates into the knowledge base for contexts in the knowledge base that have yet to be triggered. After each assumption predicate is inserted, the knowledge base is checked for inconsistencies with the newly added default knowledge. If the consistency check fails, the assumption is removed from the knowledge base. The module continues inserting **as-**

sume_CTXT predicates into the knowledge base until no contradictions are found or the set of assumptions is empty. Once this is the case, the prover reinserts the question axiom into the knowledge base and again checks for newly inferred knowledge inferred from the hypothetical. If no new inferences are derived, the module returns to assuming the preconditions of other contexts that have yet to be explored. This technique allows us to keep track of everything that has been assumed by the prover by simply examining the trace of the proof search for the **assume_CTXT** predicate. This is a very important feature of the default reasoning module because it allows us to qualify our answers with the assumptions of the contexts. It would be incorrect to state that any assertions inferred from the assumed contexts are absolute facts.

For non-hypothetical seeking questions, where a proof-by-contradiction is desirable, the same technique is applied, but now the exit condition is a successful proof-by-contradiction as opposed to a newly inferred clause.

To demonstrate the mechanisms for evaluating hypotheticals and applying defaults in a knowledge base, the proof trace for the solution to [q5] is outlined below. The trace has been divided up into sections for clarity, but the order of the sections in the original trace has been preserved. As discussed previously the inference process starts by asserting the hypothetical context assertion introduced by the question:

Assertions from the question
(1) John_NN(x1) & _s_POS(x2,x1)& train_NN(x2)& run_VB(e1,x2,x50)& -on_time_RB(v1,e1)& after_IN(e1,e2)& depart_VB(e2,x2,x6)& Union_NN(x4)& Station_NN(x5)& nn_NNC(x6,x4,x5)& before_CTXT(e2,e1) & source_CTXT(e2,x6)

The inference engine analyzes context **c1**, **c2**, and **p1** and is unable to infer anything from these contexts until they are triggered:

Conditional Context (c1)
(2) -conditional_CTXT(c1,e1) | (trip_NN(x4) & three-hour_JJ(v2,x4) & get_VB(e2,x4,x5) & him_PRP$(x5) & there_RB(v3,e2) & on_time_RB(v4,v3) & AGT_SR(x2,e1) & AGT_SR(x4,e2) & Time_Length(t1,3,0,0) & duration_CTXT(x4,t1))

Conditional Context (c2) & Planning Context (p1)
(3) -conditional_CTXT(c2,e1) | -planning_CTXT(p1,e3) | (call_VB(e4,x1,x6) & Bill_NN(x6) & push_back_VB(e5,x1,x7) & appointment_NN(x7) & two_NN(x9) & hour_NN(x10) & nn_NNC(x11,x9,x10) & _quantity_NE(x11) & AGT_SR(x2,e4) & AGT_SR(x2,e5) & TMP_SR(x11,e5) & during_CTXT(x7,t2) & time_CTMP(t2,2005,03,15,14,0,0))

Once the trigger rule for **conditional_CTXT(c2,e1)** is able to fire, the inference process moves into context **c1** while still remaining outside of the planning context **p1**:

Trigger Conditional Context
(4) -John_NN(x1) | -_s_POS(x2,x1) | -train_NN(x2) | -run_VB(e1,x2,x50) | on_time_RB(v1,e1) | conditional_CTXT(c2,e1)
(5) on_time_RB(v1,e1) | conditional_CTXT(c2,e1)
(6) conditional_CTXT(c2,e1)

Since no assertion or refutation of John's intentions can be found, the **assume_CTXT(p1)** predicate is inserted into the search space so that the prover may enter the planning context. The detection of the **assume_CTXT(p1)** predicate also informs us that whatever is inferred from this context must be qualified by the assumptions of the context:

Trigger Planning Context
(7) -call_VB(e4,x1,x6) | -push_back_VB(e5,x1,x7) | -appointment_NN(x7) | -two_NN(x9) | -hour_NN(x10) | -nn_NNC(x11,x9,x10) | planning_CTXT(p1,e3)
(8) assume_CTXT(p1)
(9) -assume_CTXT(p1) | planning_CTXT(p1,e3)
(10) planning_CTXT(p1,e3)

After entering the planning context a solution to the question can be inferred:

Inferred Predicates
(11) time_CTMP(t2,2005,03,15,14,0,0) & during_CTXT(x7,t2) & TMP_SR(x11,e5) & AGT_SR(x2,e5) & AGT_SR(x2,e4) & _quantity_NE(x11) & nn_NNC(x11,x9,x10) & hour_NN(x10) & two_NN(x9) & appointment_NN(x7) & push_back_VB(e5,x1,x7) & Bill_NN(x6) & call_VB(e4,x1,x6)

John intended to call Bill and push back the appointment two hours.

The solution to [q9] functions similarly to the solution to [q5] without the nested planning context. In this case, the train's on time arrival will trigger **conditional_CTXT(c1,e1)** which will cause everything within the conditional context **c1** to be inferred instead of the contents of context **c2**.

The three-hour trip will get him there on time.

Discussion

Effective comprehension of the travel scenario requires a question answering system to handle:

1. Discourse level coreference resolution
2. Semantic relation and Context boundary detection
3. Discourse level temporal coreference and time stamp propagation
4. Axioms on demand such as lexical chains, semantic relation chains, and eventualities
5. Context trigger rules
6. Default reasoning

The accurate detection of contexts, semantics, triggers, and defaults is an open research problem. And do to this, the focus of this paper is on the knowledge representation and reasoning tools necessary for contexts, rather than on

detection issues. To achieve this end we made simplifying assumptions that the above issues were handled by external components. These components would serve as input to the reasoning module in the question answering system, since LCC has ongoing efforts to tackle these problems. But, to compensate for the inevitability of an incomplete knowledge base and imperfect semantic and context detection, we continue to use backoff strategies as outlined in (Moldovan et al 2003).

Conclusion

Recognizing, representing, and reasoning within context, a question answering system is much more effective at retrieving accurate and precise results as demonstrated in (Moldovan et al 2005). The motivating examples detailed in the previous sections reinforce this notion by establishing that contexts and default assumptions are critical in retrieving information for user queries. It is clear that partitioning and storing content by context allows a question answering system to instantly bypass large portions of its knowledge repository and focus on generating contextually relevant inferences. Further, it provides a means for dealing with knowledge that appears contradictory when it is processed in a single, contextually insensitive, plane. But most importantly, contextual reasoning prevents the system from making mistakes by only providing answers that meet the contextual constraints of the question. No longer will answers be provided "out of context", since assertions that are true in one setting are not necessarily true in another. Additionally, defaults serve as important nuggets of information when the context constraints of the text source are overly strict.

Acknowledgments

This work was funded in part by the ARDA AQUAINT program. Additionally, LCC would like to thank Abraham Fowler and Bob Hauser for their contributions to the section "A Semantically and Contextually Rich Knowledge Representation".

References

James Allen. 1991. *Time and Time Again: The Many Ways to Represent Time*. International Journal of intelligent Systems, July 1991, pp341-355.

David Bixler, Dan Moldovan and Abraham Fowler. 2005. *Using Knowledge Extraction and Maintenance Techniques to Enhance Analytical Performance*. Proceedings of 2005 International Conference on Intelligence Analysis.

Doug Lenat. 1998. *The Dimensions of Context-Space*. CYCORP Internal Report.

John McCarthy. 1993. *Notes on Formalizing Context*. Proceedings of 1993 International Joint Conference on Artificial Intelligence.

Dan Moldovan, Christine Clark, Sanda Harabagiu 2005 *Temporal Context Representation and Reasoning.* To Appear in the Proceedings of the International Joint Conference on Artificial Intelligence 2005.

Dan Moldovan, Christine Clark, Sanda Harabagiu, Steve Maiorano 2003 *COGEX: A Logic Prover for Question Answering.* Proceedings of the Human Language Technology and North American Chapter of the Association for Computational Linguistics 2003, pages 87-93.

Ian Niles and Adam Pease. 2001. *Towards a Standard Upper Ontology*. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems. Ogunquit, Maine, October 2001

Vasile Rus and Dan Moldovan. 2002 *A High Performance Logic Form Transformation.* International Journal on Artificial Intelligence Tools, Vol. 11, No. 3 (2002) 437-454

Larry Wos. 1998. *Automated Reasoning, 33 Basic Research Problems*. Prentice Hall, 1988.