An Architecture for Real-time Reasoning and Learning

Pei Wang, Patrick Hammer, and Hongzheng Wang

Department of Computer and Information Sciences, Temple University, USA
{pei.wang,patrick.hammer,hongzheng.wang}@temple.edu

Abstract. This paper compares the various conceptions of "real-time" in the context of AI, as different ways of taking the processing time into consideration when problems are solved. An architecture of real-time reasoning and learning is introduced, which is one aspect of the AGI system NARS. The basic idea is to form problem-solving processes flexibly and dynamically at run time by using inference rules as building blocks and incrementally self-organizing the system's beliefs and skills, under the restriction of time requirements of the tasks. NARS is designed under the Assumption of Insufficient Knowledge and Resources, which leads to an inherent ability to deal with varying situations in a timely manner.

1 Various Versions of "Real-time"

Roughly speaking, solving problems "in real-time" means there are time requirements on the problems to be solved, coming out of the needs or restrictions of the domain. This topic has both theoretical and practical significance, because traditional theories of computation do not take it into consideration, while many (if not all) practical applications have time requirements.

In computability theory [6], the time spent in a computation is not considered, as far as it is finite. In computational complexity theory [1], processing time is usually considered as an attribute of an algorithm (a solution of a problem), rather than of a problem itself. Furthermore, in algorithm analysis, the "time complexity" of an algorithm indicates how fast its processing time increases as the size of problem instances, but not the actual period of time costed by the process, since that not only depends on the algorithm, but also on the size and content of the instance, the processing speed of the (hardware and software) platform, etc.

Therefore, the above theories do not cover problems that have concrete time requirements as part of their specifications. The most common form of time requirement is a deadline, which is also called "hard real-time". The other form, "soft real-time", correspond to the situation where the utility of the solution is a decreasing function of time [15, 10, 11]. We can see the former as a special case of the latter, where a deadline is the point where the utility of the solution drops from 1 to 0.

From the perspective of Artificial General Intelligence (AGI), "to work in real-time" can be taken in a more general form, including the following aspects:

– New problems can appear at arbitrary time, rather than only when the system is idly waiting for them.

- The time requirement of a problem instance can change in its different occurrences, that is, the same problem instance may have different urgency levels in different cases.
- The time requirement of a problem instance can change during its processing, that is, to become more or less urgent, rather than fully known at the beginning of the process.
- The relevant data or knowledge used in problem-solving comes incrementally after the solving process starts, partly due to the active learning and exploring of the system.
- It is desired for the best-so-far solutions to be reported whenever they are found, even though they may need to be revised or updated later.

Real-time problem-solving is not a new topic at all, though there is still no theory or technique that provides a general solution. The major approaches explored in artificial intelligence and computer science include the following:

- To find a problem-specific design by considering all software-hardware factors to meet specific time requirements in one concrete application, though parameters in the design allow flexibility of using the design in similar applications [9];
- To depend on a meta-level reasoning process to find a proper solution among the given candidates by considering their quality and time requirement, so as to get the best balance between them according to the current requirement [7, 13, 3];
- To use an "anytime algorithm", i.e., an interruptible procedure that incrementally updates the best-so-far solution, to achieve a flexible quality-time trade-off [2, 22];
- To share processing power among multiple tasks as in an operating system, where tasks are prioritized to reflect their levels of [14]. Such a system can accept a new task in any moment (assuming sufficient memory), processes it with a speed according to its priority, while guaranteeing to avoid starvation.

Though each of the above techniques addresses certain forms of time requirements, and has its applicable domains, none of them has handled all the aspects of real-time in the AGI context as mentioned previously, so a new architecture is in demand. In the next section, we will introduce NARS and how it satisfies the relevant demands.

2 NARS as a Real-time System

NARS, standing for Non-Axiomatic Reasoning System, is a general-purpose AI project [17, 20]. Many of the topics addressed in the following have been described in the previous publications [16, 18, 19, 5], though this paper is the first time when the issue of real-time in NARS is comprehensively discussed.

NARS is designed according to the opinion that intelligence is adaptation under AIKR (the Assumption of Insufficient Knowledge and Resources), meaning that the system has finite information processing capability, while it has to work in real-time, and be open to novel tasks [21]. On the other hand, as an AGI system, NARS is supposed to deal with arbitrary problems which can occur at arbitrary time. There is no guaranty that the system has enough data and an appropriate algorithm, or can predict every

possible variation of the situation. So the AIKR principle is necessary and satisfactory. Therefore, "to work in real-time" is a fundamental design requirement of NARS, and interpreted in a very broad sense that covers all aspects listed in the previous section.

As a new task can show up at any moment with a time requirement, and may be novel to the system, it means that the system cannot process it by following an existing routine or algorithm, but has to construct a solution at run time, in a case-by-case manner, for the problem (instance, not type) [18]. It also means there is no way to guarantee solutions of a fixed quality. Instead, the system simply does its best under the knowledge–resource restriction, evaluated globally with respect to all the existing tasks.

Concretely speaking, a task for NARS to carry out may be a piece of new knowledge to be absorbed into the system's beliefs, a question to be answered according to the relevant beliefs, or a goal to be achieved by executing relevant operations of the system. Under AIKR, the system has no algorithm that describes the complete process for a task, and nor can the system compare all solutions then pick the best. Furthermore, as new tasks constantly come to the system, the processing of a task may be interrupted or terminated at unanticipated moments by other more urgent tasks.

As a reasoning system, NARS processes tasks using its beliefs. In each inference step, a task interacts with a belief, and that may lead to a partial solution for the task, and at the same time generates derived tasks. Given the assumption of insufficient knowledge, no task can be "fully resolved" in a predetermined number of steps, though more steps usually lead to better solutions. On the other hand, given the assumption of insufficient resources, normally no task can interact with all relevant beliefs in the system, so as to reach a "logical end" and no better solution can be found.

Working in such a situation, the system is made to work in real-time by making each inference rules to only cost a small constant time, and allowing the processing of a task to stop after any number of inference steps.

This strategy can only be used with a logic that is fully compatible with AIKR. The Non-Axiomatic Logic (NAL) used in NARS [20] satisfies this requirement. All inference rules of NAL are "local" in the sense that the conclusion is only generated and justified with a small number of (usually one or two) premises. All non-local effects in the system are produced by multi-step processes. Consequently, the time granularity of atomic (uninterruptible) activity in NARS is very small (currently below milliseconds).

In this way, the actual solution obtained for a task is by the beliefs interacting with it (which decides the inference rules triggered in a data-driven manner), as well as their order. All these decisions are made at run time when the task is processed, rather than planned in advance.

To achieve the overall efficiency in resource allocation, the selections of tasks and beliefs are made in a specially designed data structure called "bag" [16], which is basically a probabilistic priority queue with a *put* and *take* operation (see Fig. 1) for the adding of elements and their retrieval. Each data item in a bag has a *priority* value attached, which is positively correlated with the probability for the item to be selected (with *take*) in the next round. The priority also decides which element to remove when a new item is added via *put* at full capacity, namely the lowest priority item. This makes sure finite space constraints are maintained in the "bag".

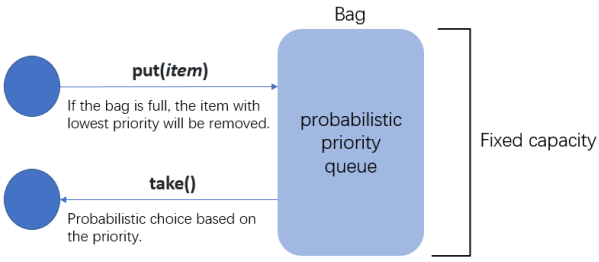

Fig. 1. The basic functions of a bag.

The priority of an item is a summary of a number of factors, including its quality, usefulness in history in similar contexts, relevance to the current situation, etc., and is dynamically adjusted according to the feedback of its usage and the change in the environment. There is also an across-the-board forgetting mechanism that decreases all priority values over time. Different items have different forgetting rate, which is indicated by a *durability* value. There is also a *quality* value, which shows the long-term significance of a data item to the system. This *priority–durability–quality* triple forms the "budget" of a task (or belief), which summarizes its relative competitiveness in the system's resource allocation at the moment.

When a user assigns a task to NARS, an initial budget can be provided, otherwise system defaults will be given according to the type and features of the task. After that, the system will adjust the budgets, as well as to decide the budget for each derived tasks and beliefs by taking the relevant factors into account.

From the viewpoint of the users, the system processes multiple tasks in a time-sharing manner, though here the mutual interference among the tasks is much stronger than that among the processes in an operating system. For a task, its processing path and results not only depend on its budget and the existing beliefs, but also on the budgets of the coexisting tasks, as well as their processing paths.

When the processing of a task stops, usually it is not because the process has reached its final state or the quality of the solution has reached a certain criterion, but because the task has lost in the resource competition, though the processing may be resumed at a future time.

Beside the above automatic resource-allocation mechanism, NARS also supports more complicated time management. For instance, the system can have knowledge about the execution time of an operation or operation sequence, and can use it in a planning process to decide whether to use the operation to reach a certain goal. For instance, to meet a specific deadline, careful planning and scheduling will be needed.

The above architecture and mechanism makes NARS capable of working in real-time, in a manner similar to that of a human in similar situations.

3 Examples

The following examples serve as an illustration of the discussed real-time aspects of NARS on what it means to operate in real-time. These examples can directly be tested with the current version of OpenNARS[1], an open-source implementation of NARS.

New information coming in while solving a problem

After a detective presents initial information and relevant background knowledge, the system changes its mind about who is the murder after newly presented evidence.

```
//It is known that the first suspected murder,
//Rambo is living in NYC and known to be aggressive
<{Rambo} --> (&,(/,liveIn,_,NYC),[aggressive])>.
//It is known that the other
//suspected murder lives in Philadelphia
<{Sam} --> (/,liveIn,_,Philadelphia)>.
//Also it is known from a psychological study that
//murder tend to be more aggressive to some degree
<murder --> [aggressive]>. %0.7%
//Who is the murder?
<{?who} --> murder>?
//System considers Rambo more likely to be the murder
Answer <{Rambo} --> murder>. %1.00;0.22%
//24 days later...
24000
//the detective got the information, that the
//murder surely must be from Philadelphia
<murder --> (/,liveIn,_,Philadelphia)>.
//Now the system thinks Sam is more likely the murder:
Answer <{Sam} --> murder>. %1.00;0.45%
```

The same order of finding solutions could possibly also have happened if the new information would have been known from the beginning. But due to the low truth-value obtained from the psychological study, and the control mechanism's tendency to pursue more truthful paths of reasoning, it is more likely that it would have found the right solution first. Note that the problem-solving process here is similar to that of an anytime algorithm, with the difference that in which order the solutions are found is not deterministic, and that not all information is demanded to be present at the beginning.

Event sequences

OpenNARS' ability to stay responsive to new incoming event sequences

```
//A sequence of entities are observed
//a was observed
<{a} --> [observed]>. :|:
251
//b was observed
<{b} --> [observed]>. :|:
...
//g was observed
<{g} --> [observed]>. :|:
131
//h was observed
<{h} --> [observed]>. :|:
```

[1] http://opennars.org/

```
//What comes after c?
<(&/,<{c} --> [observed]>,?i) =/> <{?what} --> [observed]>>?
//d comes after c
Answer <(&/,<{c} --> [observed]>,+232) =/>
                  <{d} --> [observed]>>. :-1088: %1.00;0.44%
//What comes before f?
<(&/,<{?what} --> [observed]>,?i) =/> <{f} --> [observed]>>?
//e comes before f
Answer <(&/,<{e} --> [observed]>,+602) =/>
                    <{f} --> [observed]>>. :-239: %1.00;0.43%
```

This example illustrates the system's ability for event processing. The key here is that the processing time for a new event does not increase with the amount of events seen so far. This means the system is guaranteed to stay responsive to new input, while of course it cannot be guaranteed that every possible pattern will be extracted. The system's control mechanism tends to recognize patterns that span a relatively shorter time distance, are truthful, repeating, goal-relevant, conceptually important etc. (see [4] and [5] for more details). However, if relevant questions arise, the control mechanism can make question-driven inference, as to answer a specific question which answer otherwise would have less likely been generated by the system. Example:

```
//Does h come after a?
<(&/,<{a} --> [observed]>,?1) =/> <{h} --> [observed]>>?
//h comes after a
Answer <(&/,<{a} --> [observed]>,+1789) =/>
                      <{h} --> [observed]>>. %1.00;0.09%
```

Sensitivity on elapsed processing time

This example shows the system's tracking capability of elapsed time.

```
//Lighting usually generates thunder in 5 seconds
<(&/,<lighting --> [seen]>,+500) =/>
                        <thunder --> [heard]>>. %1.0;0.45%
//Lighting is seen right now
<lighting --> [seen]>. :|:
//10 seconds later:
1000
//Do you hear thunder?
<thunder --> [heard]>? :|:
//I should have heard a thunder 5 seconds ago
Answer <thunder --> [heard]>. :-506: %1.00;0.40%
```

Many existing techniques, even when a deadline is taken into account, do not assume that the processing itself leads to the passing of problem-relevant time duration. In the above example we see that the answer to the question, which originally was a prediction, is already an event of the past. This is captured by the occurrence time of the prediction the system tracked, which in this example is, after the additional 10 seconds passed, already smaller than the current time. Generally, the system tries to find answers which are both more reliable and closer to the occurrence time of the question. Also, for decision making, the system tries to use procedure knowledge which preconditions were fulfilled more recently by events, these tend to be still valid to base a decision on. Here, both the occurrence time and truth-value are taken into account.

4 Comparisons and Discussions

Overall, there are the following possibilities for real-time problem solving:

- Using a single program specially designed to meet a predetermined time restriction under all circumstances.
- Selecting a program among an existing set of programs according to their running time and the current time demand.
- Constructing a program following a meta-algorithm according to the time requirement in the program specification.
- Running an interruptible program that has a repeatable path but unpredictable ending. It has no fixed complexity, but has a time–quality function.
- Building a one-time procedure according to time requirement of the problem and the context, without accurate predictability and repeatability.

One type of common and widely used real-time system is the programs used for the control in automation, e.g. the program for a mechanical arm to assemble a product. Such a program is relatively simpler than other kind of approaches because the environment of such program is always well defined and stable, and it is also customized for a single process. So the mechanism can run smoothly following the determined schedule. Obviously, such specifically designed program cannot deal with unexpected events, which appear in many situations.

Some other approaches are more flexible to deal with various types of deadline. Design-to-time [3] approach prepares multiple solutions and organizes the solution by making trade-offs between quality and time. An anytime algorithm [2] can be interrupted at any moment to get the most satisfied solution within that deadline, so is closer to our expectations because it can still get a satisfactory solution even if the deadline is unexpectedly changed, assuming it already found a solution to be refined further. The control mechanism of NARS shares this property, which makes it possible for the system to operate without knowing relevant deadlines beforehand. Compare to anytime algorithms, the mechanism in NARS excludes not only the predetermined final states, but also the predetermined path of processing. As long as an approach is based on a predetermined algorithm, it cannot handle unexpected changes in the environment during its running.

A fundamental difference between NARS and many traditional theories of intelligence is the AIKR principle. Under AIKR, any belief can be changed according to new information. NARS does not assume any absolutely certain knowledge about the future, and it allows unexpected changes to occur at any time. Therefore, NARS is prepared to respond to new input during the process and adjust its beliefs and inference paths. In this paper, we regard "real-time" as a more general situation. We regard the time as a part of problem, so there is fundamental difference between the idea of "real-time" and computational complexity theory. Since we cannot guarantee the environment is always suitable for a prepared algorithm, AIKR makes the NARS approach necessary to handle such situations.

While some real-time systems have been designed to take insufficient resources into account (such as schedulers in operating systems), attacking both insufficient resources and knowledge raises additional challenges. The NARS approach is somehow like the

student doing the programming question with limited memory and run time, but doesn't have the ability to deal with this question perfectly, then how to get the imperfect but proper result with such restriction.

The ability to work under AIKR is significant for a general purpose AI system, especially if real-time responses should be supported. There are also other AGI researches which apply similar principles. In the Anytime Bounded Rationality (ABR) model [12], knowledge is bounded within fixed memory budget and can be updated and revised based on the new experience, while the ABR model schedule inferences deterministically using objective time semantics. The Economic attention allocation (ECAN) model of OpenCog [8] also considers the space limitation and applies two key parameters, STI (short-term importance) and LTI (long-term importance), to manage the resource allocation. However, ECAN does not stress real-time operating conditions as the ones discussed in this paper.

In Section 2, we describe how NARS works in real-time. The system solves problems in a case-by-case manner using procedures composed at run-time, by taking many factors in the current context into account, rather than following a predetermined algorithm for that type of problem. Therefore, there is no specific requirement to the environment. NARS does not guarantee the quality and delivery time of solutions. If more knowledge and resource are provided, NARS may obtain a better solution. If the process is interrupted, NARS can still provide the best solution under existing knowledge and resources which was found, for instance by returning the highest-confident candidate solution found so far. Without the restriction of needing a predetermined algorithm for the problem to solve, NARS is more flexible in various situations, as also argued in [19].

It is however also clear that in some applications, flexibility is not the major factor under consideration. For instance in real-time operating systems, time requirements are sometimes known (example: let programs running on it respond to I/O within 1 millisecond, and under all circumstances), and can be taken into account by the corresponding specifically designed scheduling strategy. In this case, the specifically designed algorithm will of course be superior to NARS, but the algorithm cannot be used when time requirements are not known beforehand.

5 Conclusion

We have seen that "real time" has different interpretations, and that NARS fulfills multiple requirements typical for real-time systems, also going further in certain aspects. Some of the requirements we believe to be most relevant have been demonstrated with examples in Section 3: the ability to accept a new problem and new information while still working on previous ones (staying open), the resource demand for processing new events not being dependent on the amount of events seen so far (staying responsive), and the system's sensitivity to elapsed time while processing tasks, incorporating elapsed time in question answering and decision making.

Especially since the system does not assume the existence and knowing of deadlines, it fulfills more than certain common understandings of the term "real-time" would ask for. However, since it makes no guarantee about the quality of the solutions to be

found, or even to find one at all, it is also in some sense "weaker" than the alternative techniques. This weakness, as argued, follows from the fundamental restriction of operating under Insufficient Knowledge and Resources. The requirement of real-time responses is often satisfiable only by the re-allocation of available resources, as a direct consequence of the often insufficient computational resource supply. Also, as argued, the system cannot make any guarantees about the quality of its solutions, also due to the often insufficient knowledge it has available to solve problems.

Aspects coming from AIKR shed some light on what we believe to be unavoidable properties of AGI systems, unless inherently different philosophies are followed. Philosophies with views like computational resources are infinite, and/or the system does always know the problem-relevant information, assume too much to be acceptable. As we argued, an AGI should work in any environment, including uncertain ones as well. Hence AIKR cannot be dropped, and leads to systems falling in its own subcategory of "real-time system" to be studied further.

Acknowledgement

This work is partially supported by a gift from the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, 3rd edn. (2009)
2. Dean, T., Boddy, M.: An analysis of time-dependent planning. In: Proceedings of AAAI-88. pp. 49–54 (1988)
3. Garvey, A., Lesser, V.: Design-to-time Real-Time Scheduling. IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Planning, Scheduling and Control 23(6), 1491–1502 (1993)
4. Hammer, P., Lofthouse, T.: Goal-directed procedure learning. In: Proceedings of the Eleventh Conference on Artificial General Intelligence. pp. 77–86 (2018)
5. Hammer, P., Lofthouse, T., Wang, P.: The OpenNARS implementation of the Non-Axiomatic Reasoning System. In: Proceedings of the Ninth Conference on Artificial General Intelligence. pp. 160–170 (2016)
6. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston, 3rd edn. (2007)
7. Horvitz, E.J.: Reasoning about beliefs and actions under computational resource constraints. In: Kanal, L.N., Levitt, T.S., Lemmer, J.F. (eds.) Uncertainty in Artificial Intelligence 3, pp. 301–324. North-Holland, Amsterdam (1989)
8. Ikle, M., Pitt, J., Sellmann, G., Goertzel, B.: Economic attention networks: Associative memory and resource allocation for general intelligence. In: Proceedings of the Second Conference on Artificiel General Intelligence. pp. 73–78 (2009)
9. Korf, R.E.: Real-time heuristic search. Artificial Intelligence 42(2-3), 189–211 (1990)
10. Laffey, T.J., Cox, P.A., Schmidt, J.L., Kao, S.M., Read, J.Y.: Real-time knowledge-based systems. AI Magazine 9, 27–45 (1988)
11. Musliner, D.J., Hendler, J.A., Agrawala, A.K., Durfee, E.H., Strosnider, J.K., Paul, C.J.: The challenges of real-time AI. Computer 28(1), 58–66 (Jan 1995)
12. Nivel, E., Thórisson, K.R., Steunebrink, B., Schmidhuber, J.: Anytime bounded rationality. In: Proceedings of the Eighth Conference on Artificial General Intelligence. pp. 121–130. Springer (2015)
13. Russell, S., Wefald, E.H.: Principles of metareasoning. Artificial Intelligence 49, 361–395 (1991)
14. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts. Wiley Publishing, 9th edn. (2012)
15. Stankovic, J.A.: Real-time computing systems: The next generation. Tech. Rep. 88-06, University of Massachusetts, Amherst (1988)
16. Wang, P.: Problem solving with insufficient resources. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems 12(5), 673–700 (2004)
17. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006)
18. Wang, P.: Case-by-case problem solving. In: Proceedings of the Second Conference on Artificial General Intelligence. pp. 180–185 (2009)
19. Wang, P.: Solving a problem with or without a program. Journal of Artificial General Intelligence 3(3), 43–73 (2012)
20. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)
21. Wang, P.: On defining artificial intelligence. Journal of Artificial General Intelligence 10(2), 1–37 (2019)
22. Zilberstein, S.: Operational rationality through compilation of anytime algorithms. AI Magazine 16(2), 79–80 (1995)