Anytime Bounded Rationality

Eric Nivel[1], Kristinn R. Thórisson[1,2], Bas Steunebrink[3], and Jürgen Schmidhuber[3]

1 Icelandic Institute for Intelligent Machines
2 Reykjavik University, CADIA
3 The Swiss AI Lab IDSIA, USI & SUPSI

Abstract. Dependable cyber-physical systems strive to deliver *anticipative*, *multi-objective* performance *anytime*, facing deluges of inputs with *varying* and *limited* resources. This is even more challenging for life-long learning rational agents as they also have to contend with the *varying* and *growing* know-how accumulated from experience. These issues are of crucial practical value, yet have been only marginally and unsatisfactorily addressed in AGI research. We present a value-driven computational model of anytime bounded rationality robust to variations of both resources and knowledge. It leverages continually learned knowledge to anticipate, revise and maintain concurrent courses of action spanning over arbitrary time scales for execution anytime necessary.

1 Introduction

Key among the properties mission-critical systems call for is *anytime* control – the capability of a controller to produce control inputs whenever necessary, despite the lack of resources, trading quality for responsiveness [3,5]. Any practical AGI is constrained by a mission, its own architecture, and limited resources including insufficient time/memory to process all available inputs in order to achieve the full extent of its goals *when it matters*. Moreover, unlike fully hand-crafted cyber-physical systems, AGIs should handle underspecified dynamic environments, with no other choice but to *learn* their know-how, possibly throughout their entire lifetime. The challenge of anytime control thus becomes broader as, in addition to resource scarcity, it must encompass inevitable variations of completeness, consistency, and accuracy of the learned programs from which decisions are derived.

We address the requirement of delivering *anticipative*, *multi-objective* and *anytime* performance from a *varying body of knowledge*. A system must anticipate its environment for taking appropriate action – a controller that does not can only react after the facts and "lag behind the plant". Predictions and sub-goals must be produced concurrently: (a) since achieving goals needs predictions, the latter must be up to date; (b) a complex environment's state transitions can never be predicted entirely: the most interesting ones are those that pertain to the achievements of the system's goals, so these must be up to date when predictions are generated. A system also needs to achieve multiple concurrent goals to reach states that can only be obtained using several independent yet temporally correlated and/or co-dependent courses of action while anticipating and resolving potential conflicts in due time. The capabilities above must be leveraged to compute and *revise* plans continually, as resources allow and knowledge accumulates, and execute them whenever necessary, as situations unfold – this requires subjecting a system's deliberations (and execution) to deadlines relative to an external reference (world) clock.

Most of the strategies controlling the life-long learning AI systems we are aware of are subject to one or several severe impediments to the responsiveness and robustness we expect from mission- and time- critical systems. First, a sequential perception-decision-action cycle [1,6,7,8,12] limits drastically the potential for situational awareness and responsiveness: such "cognitive cycles" are difficult to interrupt and, being driven by *subjective* inference "steps", are decoupled from *objective* deadlines either imposed or learned. Second, interleaving multiple trains of inference in one sequential stream [1,4,6,7] results in the overall system latency adding up with the number of inputs and tasks at hand: such a system will *increasingly* and irremediably lag behind the world. Third, axiomatic reasoning [1,6,7] prevents the revision of inferences upon the acquisition of further amounts of evidence and know-how, prohibiting continual refinements and corrections. Last, the lack of explicit temporal inference capabilities [1,6,7,8] prevents learned procedural knowledge from inferring deadlines for goals and predictions, which is needed to plan over arbitrary time horizons – on that front, state-of-the-art reinforcement and evolutionary learners [9,13,2] present other inherent difficulties. NARS [14] notably avoids these pitfalls and could, in principle, learn to couple subjective time semantics to a reference clock and feed them to a probabilistic scheduler. We set out instead to schedule inferences deterministically using objective time semantics so as to avoid the unpredictability and unreliability that inevitably arise from using inferred time semantics to control the inferencing process itself.

We present a computational model of anytime bounded rationality (we refer to this model as ABR) that overcomes the limitations above. It posits (a) a dynamic hierarchy of revisable time-aware programs (called *models*) (b) exploited by concurrent inferencing jobs, that are (c) continually re-scheduled by (d) a value-driven executive under (e) bounded latencies, keeping the size of all data (inputs, inferences, programs and jobs) within (f) a fixed memory budget. This model has been implemented and tested: it constitutes the *control* core of our auto-catalytic, endogenous, reflective architecture (AERA; [10]), demonstrated to learn (by observation) to conduct multimodal interviews of humans in real-time, bootstrapped by a minimal seed [11]. While the learning algorithm of this system has been described in prior publications [10,11], its control strategy, the ABR model, has not been published elsewhere.

2 Overview of ABR Control

Our anytime bounded rationality model (ABR) assumes (Fig. 1) an executive (in black), a memory (dashed areas) and a set of I/O devices (dedicated external sub-systems). Programs include *monitors* (to assess the outcomes of goals and predictions) and *models*[1]. Models are either hand-crafted or learned from experience, and present

[1] Other programs construct new models from life-long experience, see [10] for details.

varying degrees of consistency, accuracy, and reliability. Jobs are requests for processing one input by one program and are executed by a pool of threads; as a result of job execution, new inputs and programs are added to the system, other programs are deleted and some inputs cancelled. ABR is data-driven: a writer (`W`) creates new jobs upon matching inputs to programs while an antagonist eraser (`E`) enforces a forgetting strategy to limit memory usage. Inputs consist of sensory inputs, *reflective* inputs (traces of model execution, see section 3), inferences and drives (drives are user-defined top-level goals and constraints). Outputs are commands executed asynchronously by the I/O devices: these respond with *efference copies* (considered sensory inputs) telling the system what has *actually* been executed (and when), as opposed to what was *intended*, thus allowing it to learn (i.e. to model) the devices' behaviors.

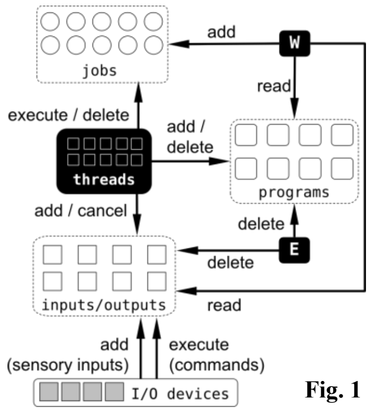

Fig. 1

Models produce revisable inferences in two modes, forward chaining (predictions) and backward chaining (sub-goals). Both are performed *concurrently* – a model can produce several predictions from several different inputs while at the same time producing several sub-goals from several other goals (section 3). Motivated by drives, a hierarchy of models produces *concurrent overlapping* cascades of simulated sub-goals – at the bottom of a cascade, terminal goals embed commands to I/O devices, executed when such goals are committed to. Goal cascades simulate alternate courses of actions to achieve *multiple* (possibly) *concurrent goals*. These projected plans are continually re-evaluated upon knowledge updates (addition/deletion of inputs, inferences and models), and maintained for execution anytime it matters: continually anticipating the expiration of simulated goals' deadlines, a system is pressed to commit to these goals (and their ancestors), comparing their value (and their ancestors') to that of other conflicting or redundant goals in order to enact the best actions planned so far. In parallel to these top-down simulations, the model hierarchy is traversed by bottom-up concurrent overlapping flows of predictions originating from I/O device readouts. These warn the system of the predictable success or failure of its goals, and prompt it to adapt its behavior anticipatively by considering alternate goals, producing new ones and/or increasing or decreasing the importance of existing ones, possibly downplaying some (section 4).

Jobs are assigned a *priority* that determines the time when they may be executed. Priorities are *continually updated*, thus allowing, at any time, high-value new jobs to get executed before less important old jobs, and old jobs to become more valuable than newer ones based on new evidence. Jobs of lesser priority may get delayed repeatedly and eventually cancelled, as is likely to happen when the system is overloaded. A job priority depends on the continual assessment of the *relevance* of the program and the *tending value* of the input (see below). Threads recompute priorities, delete jobs that have become irrelevant and pick the best jobs for execution. Such scheduling overhead is *bounded by a constant* – a thread only updates the priorities of an ever-changing (fixed size) subset of the jobs [2]. All execution times are commensurate: memory usage is proactively limited (see below), and the threads', `E`'s and `W`'s worst-case execution times (WCET) are all *identical* and *constant*.

Assuming the life-long learning of new models and a sustained influx of inputs, the number of jobs and input-to-program matching attempts can grow exponentially and exceed the memory budget. This growth is limited by a forgetting strategy based on the prediction of the amount of available memory, inferred, conservatively, from past experience – essentially, the rates of data creation and deletion (inputs, inferences, programs and jobs). Should `E` anticipate a shortage, it deletes the necessary number of data in order to accommodate the next predicted influx while preserving the most valuable existing data: the top-rated candidates for deletion are the inputs that contributed the least recently to the achievement of goals, the least reliable models that succeeded the least recently, and the jobs of the least priority.

A system's experience constitutes defeasible knowledge, and is thus represented using a non-axiomatic temporal term logic, truth being neither eternal nor absolute. A term exposes three components: (a) arbitrary data, (b) a *time interval* ([early deadline, late deadline] in microseconds, world time) within which the data is believed to hold (or, if negated, during which it is believed not to hold) – an inference's lifetime being bounded by its late deadline – and, (c) a *likelihood* (in [0,1]), the degree to which the data has been ascertained. The likelihood of a sensory/reflective input is one whereas that of a drive is user-defined. An inference results from the processing of evidences by chains of models[3], and is defeated or vindicated upon further (counter-)evidences. Its likelihood is continually revised depending on the context and reliability of said models and, notably, decreases with the length of the chains (see next section).

The value of tending to an input x (sensory/reflective input, inference or drive) at time t depends on both its *urgency* (for situational awareness) and likelihood:

$$Urgency(x,t) = 1 - \frac{THZ(x,t)}{Max_i\big(THZ(x_i,t)\big) + U}$$

$$TendingValue(x,t) = Urgency(x,t) \times Likelihood(x,t)$$

where $THZ = Max(LD(x) - t, 0)$ stands for "time horizon", LD for "late deadline", x_i being all the inputs in the system and U a system parameter meant to keep urgencies positive. Now, a goal may be achieved by other means than spending effort deriving sub-goals from it (e.g. when the environment is cooperative). The value of pursuing a goal x thus decreases with the most likely prediction of its target state:

$$P(x,t) = Max_i\big(Likelihood(p_i,t)\big),$$

$$Effort(x,t) = \begin{cases} Likelihood(x,t), & Likelihood(x,t) \geq P(x,t) \\ 1 - P(x,t), & otherwise \end{cases}$$

$$TendingValue(x,t) = Urgency(x,t) \times Effort(x,t)$$

[2] Details on the scheduling algorithm are outside the scope of this paper.

[3] Different chains may produce several equivalent inferences, albeit with different likelihoods. Threads will execute first the jobs performing the most likely of these inferences, postponing or discarding the others.

where p_i are the predictions of x's target state. The global *relevance* of a model m is the (normalized) maximum of the tending values of all its inferences $x_i(T,m)$ of type T (*Predictions* or *Goals*) that are still alive at time t:

$$UR(m,T,t) = \operatorname*{Max}_i\big(TendingValue(x_i(T,m),t)\big), Relevance(m,T,t) = \frac{UR(m,T,t)}{Max_i(UR(m_i,T,t))}$$

where m_i are the models in the system. If none of the $x_i(T,m)$ are alive, then m's relevance is computed as $\frac{Min_i(UR(m_i,T,t))}{Max_i(UR(m_i,T,t))}$, giving it a chance to execute, albeit with a minimal relative priority. Finally, the priority of a chaining job is the product of the relevance of the model m and the tending value of the input x:

$$PriorityForwardChaining(x,m,t) = Relevance(m,Goals,t) \times TendingValue(x,t)$$

$$PriorityBackwardChaining(x,m,t) = Relevance(m,Predictions,t) \times TendingValue(x,t)$$

Prediction and goal monitoring jobs enjoy the same priority as, respectively, forward and backward chaining jobs.

3 Models

Models are variable defeasible knowledge: experimental evidences trigger both their construction, deletion, and the continual revision [10], of their predictive performance:

$$Reliability(m,t) = \frac{e^+(m,t)}{e(m,t)+1}$$

where $e^+(m,t)$ is the number of successful predictions produced until any time t by a model m, and $e(m,t)$ the total number of predictions, both updated by *prediction monitors* each time a prediction fails or succeeds.

A model `M` (Fig. 2a) specifies a conjectured *causal relationship* between a left-hand term (LT) and a right-hand term (RT), i.e. patterns (`A` and `B`) featuring variables (`X`, `Y` and `Z`). When a sensory (or reflective) input or prediction `a` matches `A` (2b), `M` produces a prediction `b`, patterned after `B` where variables are bound to values assigned to variables shared by `A` or calculated as (learned) functions of values in `A` (`fwd`, embedded in the model) – in particular, time intervals are inferred this way. The forward execution of `M` (predicting an instance of the causal relationship) is reflected by a *model instance* term (`i`), interpreted as a prediction of `M`'s success (see rationale below). For each prediction, a new program, a prediction monitor (`PM(b)`), is created to assess its outcome. When a goal `b` matches `B` (2c), a sub-goal `a` is produced, patterned after `A` whose variables are bound to values shared by `B` or calculated as functions of values in `B` (`bwd`, also learned and embedded in the model). For each sub-goal, a new program, a *goal monitor* (`GM(a)`), is created to assess its outcome. Alternatively, when a sensory/reflective input matches a model's RT, an *assumption*, patterned after its LT, is

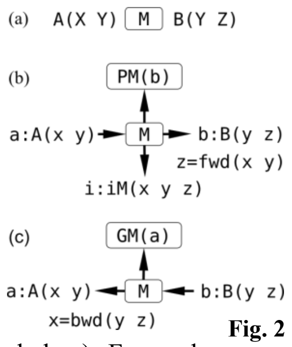

Fig. 2

produced, given that no corresponding input already matched said LT[4]. The likelihood, at any time t, of an inference y produced by a model m from an input x is:

$$Likelihood(y,t) = Likelihood(x,t) \times Reliability(m,t)$$

Note that the model instance `i`, being a prediction of `M`'s success, is assigned a likelihood equal to that of the prediction `b`.

Models form hierarchical structures: when a model $\texttt{M}_1$ features an instance of a model $\texttt{M}_0$ as its LT (e.g. $\texttt{iM}_0\texttt{(…)}$), it specifies a *post-condition* on the execution of $\texttt{M}_0$, predicting some outcome upon the successful execution of $\texttt{M}_0$, regardless of its premises; conversely, when $\texttt{M}_1$ features an instance of $\texttt{M}_0$ as its RT, it specifies a positive *pre-condition* on $\texttt{M}_0$, predicting the success of $\texttt{M}_0$ upon the occurrence of some premise. Pre-conditions can also be negative, to predict failures: in this case, the RT is of the form $\texttt{|iM}_0\texttt{(…)}$, '|' indicating failure. Pre-conditions influence the computation of the likelihood of predictions (see below) but have no impact on that of goals.

A model is called *conjunctive* (Fig. 3a) when it specifies a causal relationship whereby an effect is not entailed by one single term, but by a *context*, i.e. a set of *temporally correlated* positive pre-conditions (`*` denotes an unbound value). A conjunctive model has no LT: instead, for unification, a parameter list (`(X)`) gathers all the variables shared by positive pre-conditions unless already present in the RT (`B(Y Z)`). A conjunctive model *updates* predictions as *amounts* of (value-sharing) positive pre-conditions accumulate. Over time t, the likelihood of a prediction p produced by a conjunctive model m increases with the conjunction of positive pre-conditions weighted by their reliability, and decreases with the most likely of the negative ones:

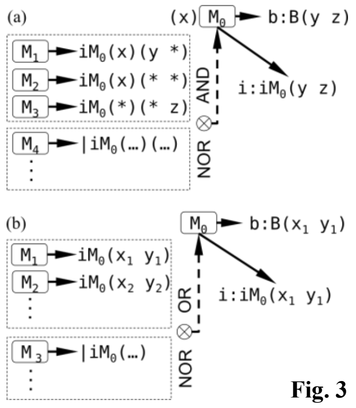

Fig. 3 Pre-conditions can be subjected to any others recursively instantiating the pictured hierarchical patterns. Logical operations are *continuous* and *persistent* instead of discrete and transient: `AND`s are weighted and compete (as well as `OR`s) with `NOR`s based on the likelihoods of pre-conditions, continually updated to reflect knowledge variations, that are both quantitative (likelihood- and reliability-wise) and qualitative (new inputs, inferences and models, deletion of underperforming models, unlikely inferences and valueless old inputs).

$$PosL(m,t) = \frac{\sum_i (Likelihood(p_{m_i},t) \times Reliability(m_i,t))}{\sum_k Reliability(m_k,t)}$$

$$NegL(m,t) = Max_j(Likelihood(p_{m_j},t))$$

$$Likelihood(p,t) = PosL(m,t) \times (1 - NegL(m,t))$$

where m_i are the pre-conditions on m that predicted m's success (p_{m_i}), m_k all the positive pre-conditions on m, and p_{m_j}, m's predicted failures. Positive pre-conditions without which the effect of the model is reliably entailed are deemed irrelevant: prediction monitors will repeatedly decrease their reliability until deletion. When presented with a goal, $\texttt{M}_0$ outputs a sub-goal ($\texttt{iM}_0\texttt{(y z)}$) targeting its own (forward) operation – this sub-goal will match the

[4] Assumptions are not essential for the present discussion and will not be detailed further.

RT of its pre-conditions and trigger the production of their respective sub-goals (or negations thereof in the case of negative pre-conditions).

A model is called *disjunctive* (Fig. 3b) when it specifies a causal relationship whereby an effect is entailed by the occurrence of the *most likely* positive pre-condition, competing with the most likely negative one. Positive pre-conditions on a disjunctive model constitute a set of *options* to entail the models' success – whereas in conjunctive models they constitute a set of (weighted) *requirements*. The likelihood of a prediction is computed as for conjunctive models, except for its $PosL$ component:

$$PosL(m,t) = Max_i(Likelihood(p_{m_i},t))$$

Fig. 4 shows part[5] of an actual system (called S1; [10]) that observed (in real-time) human interactions of the general form "take a [color] [shape], put it there [pointing at some location], thank you" (and variations thereof) and learned how to satisfy its mission – hearing/speaking "thank you", depending on the assigned role (interviewee or interviewer). S1's seed contains (a) a drive `run`, (b) a model S_0 and its context {S_1, S_2}, (c) sensors monitoring the state of hands, objects and utterances (color (`col`), position (`pos`), attachment (`att`), shape (`is`), belonging (`bel`), designation (`point`), speech (`speak`)) and, (d) effectors (commands **`move`**, **`grab`**, **`release`**, **`speak`** and **`point`**).

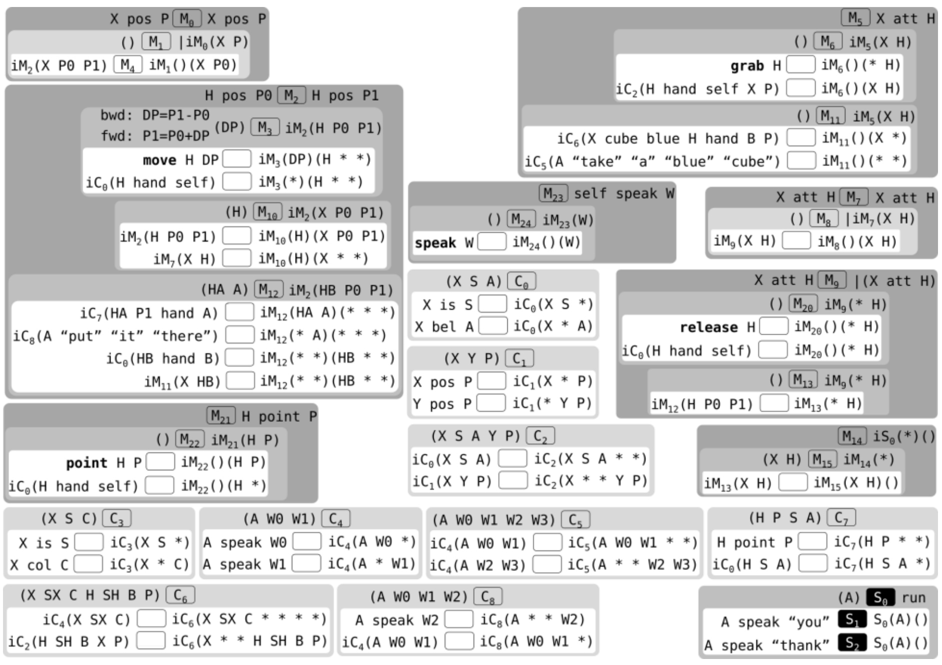

Fig. 4. Example learned model hierarchy (seed models in black).

Models M_3, M_6, M_{20}, M_{21} and M_{23} predict the consequences of issuing commands to end-effectors – they were learned by observing the results of a few randomly generated

[5] For clarity, timings and variants of learned knowledge (e.g. variations in wording, shapes and colors) are omitted. Faulty models are also omitted (see section 5).

commands ("motor babbling"). A conjunctive model specifies how a state (its RT) comes to happen when a context (white areas), i.e. a temporal correlation of pre-conditions, is observed. For example, M_{12} predicts that an object X will move when an actor B has taken it, followed by an actor A asking B to put it at a designated location. A disjunctive model subjects the occurrence of its RT state to the observation of one pre-condition among a set of options. For example, M_5 predicts that an object X will be attached to a hand H (RT of M_5) in two cases: either S1grabs the object as per model M_6, or an actor (A) asks another one (B) to take the object, as per model M_{11}. The disjunctive model M_{14} specifies how hearing "thank you" is entailed by an actor A asking an actor B to pick an object and drop it to a designated location (chain iM_{15}–iM_{13}–iM_{12}–iM_{11}). When a model features a LT, it specifies the transformation of one state (its LT) into another (its RT). Such transformations can also be controlled by pre-conditions, as for disjunctive models. For example, M_9 predicts the transition of the state "an actor holds an object X" to "X no longer held" when S1 releases the object (M_{10}), or when the actor is asked to drop the object somewhere (M_{13}). In this case, the LT has to be matched for the model to sub-goal and conversely, both the LT and one pre-condition must be observed for the model to predict. Models C_i are conjunctive models without an RT. They represent *abstractions* of sub-contexts that have been reliably identified among larger ones (those controlling conjunctive models). Occurrences of sub-contexts are encoded as model instances (iC_i) and are not subjected to any negative pre-conditions.

4 Continual Simulation & Anytime Commitment

An ABR-compliant system is multi-objective: as much as resources allow, it runs “what if” scenarios to predict the impact of the hypothetical achievement of some goals on that of other goals, anticipating conflicts and redundancies so as to commit to the best goals so far, downplaying other contenders. For each goal, a goal monitor accumulates evidences and counter-evidences of the desired state and, at the goal's late deadline, declares either a success or failure, based on the evidence (and counter-evidence) holding the greatest likelihood value – its sub-goals are cancelled and so are the corresponding chaining and monitoring jobs. A goal is either *simulated* (the default) or *actual* (defeasibly committed to). For each simulated goal, a corresponding simulated prediction is produced, used by goal monitors to evaluate the consequences of reaching the goal in question. Fig. 5 shows two simulation branches stemming from two actual goals (g_0 and g_5). The simulated achievements (grey arrows) of simulated goals (marked 's') are accumulated by the monitors of actual goals (5a). From these predictions, said monitors assess the impact (success or failure) of the simulated goals on their own goals (5b); such predicted impacts (grey dashed arrows) are in turn accumulated by the monitors of simulated goals. At the earliest of the early deadlines of the goals in a branch (say g_4's

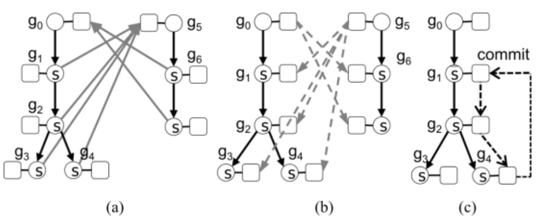

Fig. 5. Concurrent simulations and commitment.

for g_0's branch), a request for commitment is sent upward (5c) to the first simulated goal in the branch (g_1). Requests are granted depending on the predicted impact of a goal candidate (g_1) on actual goals (g_5): if g_1 entails no failure of g_5, then commit to g_1; otherwise (there is a conflict between g_1 and g_5), if g_5 is of less importance[6] than g_0, then commit to g_1 and all its sub-goals in the branch[7]; otherwise do nothing – assuming the same knowledge, the system will commit to g_6 later, following the same procedure. Commitment to g_1 is declined in case g_0 is redundant with a more important actual goal (targeting the same state).

Commitment is *defeasible*, i.e. continually revised as new knowledge (inputs, inferences and models) impact both goals' importance and tending value: after commitment, a goal monitor keeps accumulating predictions to anticipate further conflicts and redundancies that could invalidate its decision, in which case the goal (and its sub-goals) will become simulated again. When the system commits to a goal (g_1) conflicting with another one (g_5), it keeps simulating g_5 instead of deleting it, in the hope of witnessing its unexpected success, triggering the acquisition of new (possibly better) models. A system may also acquire better knowledge before g_5's deadline and uncover situations where it can be achieved without conflict – the system may then commit to some of g_5's sub-goals (e.g. g_6) without having to re-compute the simulation branches. For the same reasons, goals deemed redundant with more important ones are also kept in the simulated state.

5 Results & Conclusion

We tasked a version of our AERA architecture, S1 [10], implementing ABR, (S1; [10]) with learning to conduct natural multimodal interviews with humans. S1 learned how to do this by observing humans; an overview of the results is given in [11] from the perspective of learning. From the perspective of control, S1 learned true multi-objective control, coordinating consistently object manipulation, looking, nodding, pointing, listening and speaking. Anticipative planning over arbitrary time scales was demonstrated by S1 (a) taking turns in the interaction appropriately and in due time, (b) planning questions depending on both the interviewee's answers and time available and, (c) interrupting a talkative interviewee early in the interview to meet an imposed deadline. On the last point, pressed by this deadline, S1 planned and executed communication acts in a timely fashion, demonstrating anytime responsivity that was constrained by the timing of the humans' behaviors: interactions unfolded naturally, presenting no differences in either latencies or meaning with respect to baseline human-human interactions. S1's anytime adaptive behavior resulted from the continual development of goal simulations, regulated by concurrent and timely predictions of the humans' attention and intentions, hinted at by sequences of speech and gestures, in both form and content, over various time scales, from word and sentence utterance, to object manipulation, up to the interview's full length, thus allowing S1 to continually

[6] A goal's importance quantifies the need to reach its target state, not the need to *spend effort* reaching it, as factored in the goal's tending value. Accordingly, a goal's importance ignores predictions of the target state and is the product of the sole goal's urgency and likelihood.

[7] When a terminal goal is committed to, its command is executed by the appropriate I/O device.

reorder questions in the long term and anticipate deadline misses.

In conclusion, our model of anytime bounded rationality addresses several important issues for achieving mission-critical control in AGI-aspiring systems. It abolishes the standard cognitive cycle, and posits instead value-driven parallel competitive inferencing. Our implemented system demonstrably achieves multi-objective, anticipatory and anytime performance, under varying knowledge and resources.

Acknowledgments. This work has been partly supported by the EU-funded projects HUMANOBS (FP7-STREP-231453) and Nascence (FP7-ICT-317662), grants from SNF (#200020-156682) and Rannis, Iceland (#093020012).

References

1. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychological Review 111, 1036-1060 (2004)
2. Bellas, F., Duro, R.J., Faiña, A., Souto, D.: Multilevel darwinist brain (MDB): Artificial evolution in a cognitive architecture for real robots. IEEE Transactions on Autonomous Mental Development 2(4), 340-354 (2010)
3. Boddy, M., Dean, T.L.: Deliberation scheduling for problem solving in timeconstrained environments. Artificial Intelligence 67(2), 245-285 (1994)
4. Cassimatis, N., Bignoli, P., Bugajska, M., Dugas, S., Kurup, U., Murugesan, A., Bello, P.: An architecture for adaptive algorithmic hybrids. IEEE Transactions on Systems, Man, and Cybernetics, Part B 40(3), 903-914 (2010)
5. Horvitz, E., Rutledge, G.: Time-dependent utility and action under uncertainty. In: Proc. 7th Conference on Uncertainty in Artificial Intelligence. pp. 151-158. Morgan Kaufmann Publishers Inc. (1991)
6. Laird, J.E.: The Soar cognitive architecture. MIT Press (2012)
7. Langley, P., Choi, D., Rogers, S.: Interleaving learning, problem-solving, and execution in the ICARUS architecture. Technical report, Computational Learning Laboratory, CSLI, Stanford University (2005)
8. Madl, T., Baars, B.J., Franklin, S.: The timing of the cognitive cycle. PloS ONE 6(4), e14803 (2011)
9. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
10. Nivel, E., Thórisson, K.R., Steunebrink, B.R., Dindo, H., Pezzulo, G., Rodríguez, M., Hernández, C., Ognibene, D., Schmidhuber, J., Sanz, R., Helgason, H.P., Chella, A.: Bounded Seed-AGI. In: Proc. 7th Conference on Artificial General Intelligence, pp. 85-96. Springer, Quebec City, Canada (2014)
11. Nivel, E., Thórisson, K.R., Steunebrink, B.R., Dindo, H., Pezzulo, G., Rodriguez, M., Hernandez, C., Ognibene, D., Schmidhuber, J., Sanz, R., Helgason, H.P., Chella, A., Jonsson, G.K.: Autonomous acquisition of natural language. In: Proc. IADIS International Conference on Intelligent Systems & Agents 2014, pp. 58-66 (2014)
12. Shapiro, S.C., Ismail, H.O.: Anchoring in a grounded layered architecture with integrated reasoning. Robotics and Autonomous Systems 43(2-3), 97-108 (2003)
13. Veness, J., Ng, K.S., Hutter, M., Uther, W., Silver, D.: A Monte-Carlo AIXI approximation. Journal of Artificial Intelligence Research 40(1), 95-142 (2011)
14. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer (2006)