Chapter 6
Work on Command: The Case for Generality

Workin' 9 to 5,
What a way to make a living,
Barely gettin' by,
It's all takin' and no giving,
They just use your mind and they never give you credit,
It's enough to drive you crazy if you let it.

Dolly Parton , '9 to 5' [251]

Let us recall that, from a pragmatic perspective, AI is nothing more or less than a *tool* for implementing a new leap in *automation*. This pragmatic perspective on AI is the one that matters in the current world economy, and therefore will necessarily receive primacy for development. While we do acknowledge AI-related notions such as 'artificial life', a significant business case for 'AI as organism' has yet to be demonstrated, and therefore we consider AI that does not directly seek to deliver automation to be out of scope.

As discussed in Sect. 3.2, applications in automation with a known and well-defined target or utility function can be addressed using reinforcement learning—RL allows one to optimize a controller to perform exactly that task with guarantees on speed and accuracy. However, training policies is difficult and time consuming, and this hampers an orthogonal class of applications that require the minimization of the cost/latency incurred by engineer-then-deploy cycles. There is indeed a recurrent need in industry to streamline automation engineering and, in particular, to capitalize on learning processes, recognizing that knowledge acquired from automating one process can be useful for automating the next. Moreover, techno-economic conditions for business sustainability are shifting rapidly, as exemplified by e.g. the Industry 4.0 initiative in Europe [74, 141]: production lines need to absorb increasing levels of flexibility, meaning that processes are irreversibly moving away from the stationarity

J. Swan et al., *The Road to General Intelligence*,
Studies in Computational Intelligence 1049,
https://doi.org/10.1007/978-3-031-08020-3_6

that was once the norm. For example, a system may be tasked with controlling the assembly process of product X, but after a few months X is replaced by new-and-improved product Y, similar to X in some ways. Now we would like to tell the system:

> Stop assembling X immediately, here's a specification of Y, and here are most of your old and a few new effectors. Now start assembling Y, avoiding such and such kinds of defects and wastage.

We use the notion of 'work on command' to refer to the ability of a system to respond, at any time, to changes in task specifications, both positive (goals to be achieved) and negative (constraints to be respected). To be of any use, we include in this notion the ability to leverage all relevant knowledge from prior tasks with little effort. Such leveraging should be non-destructive, i.e., it must be possible to command the system to resume an earlier task, on which it should in general be able to perform at least as well as before.

We posit that performing work on command requires *general* intelligence. Keeping the pragmatic perspective focused on automation engineering, a system's generality can be measured as *the inverse of the cost of its deployment and maintenance in a given range of real-world task/environment spaces*.[1] It will be clear that, the more general an AI system, the better (and cheaper) it will be at performing work on command.

We have seen in Sect. 3.1 that the function of RL is to compile a policy consisting of a direct mapping from environment states to actions. In this paradigm, behavior is the (fixed) computation of a response to a stimulus, best attuned to the assumed specifications of the task and the environment. This notion of 'behavior as a curried planner' offers the benefits of speed and accuracy: once learned, a policy needs little computation to deliver optimal results. However, this comes at the cost of brittleness: adaptation is impossible should the task or environment escape the initial assumptions after deployment. But it does not have to be like this. In cybernetics, system theory, and psychology, behavior is better described as "a control process where actions are performed in order to affect perceptions" as noted by Cisek in his critique of computationalism [49]—see also von Uexküll [191]. Since it is a process, a behavior can be more easily adapted to the variability of its goal, environment, and contingencies. For this reason, we consider processes, *not algorithms*, as the concept most constructive for the synthesis of intelligent systems, as opposed to purely reactive systems—see also "The Irrelevance of Turing Machines to AI" by Sloman [316].

From this perspective, agents have to learn a world model which allows for dynamic planning through simulated trajectories.[2] The value proposition is that this

[1] One might argue that generality is better defined as inversely proportional to computational effort than to monetary cost. However, both involve empirically-determined (but in general, arbitrary) resource expenditure. From the pragmatic, real-world perspective taken here, we find money the more suitable unit of measurement.

[2] As noted before, even if model-based RL leverages world models while learning policies, the final outcome remains a fixed behavior.

model is *task-agnostic*, making it a general-purpose corpus of knowledge that can be reused for achieving a broad variety of goals and shared with other agents to hasten their learning.

The general requirement to perform work on command can be broken down in three main components:

- To handle explicit specifications of goals/constraints and make provision for their possible change.
- To plan dynamically using world models which may change *during* planning.
- To deliver plans and learn anytime, i.e., asynchronously with regards to the system activity.

These requirements add to the challenges for RL discussed previously and require to alter, significantly, its conceptual basis: in the next two sections, we describe how RL can be viewed as an 'artificially constrained' version of an extended framework which we term WoC-RL (WoC for *work on command*). The purpose of WoC-RL is purely pedagogical: it serves to introduce aspects of the subsequently described 'Semantically Closed Learning' from the familiar perspective of RL. In the last section, we depart from the ML algorithmic world view and propose a process-centric perspective on system agency to address the requirement of anytime operation.

6.1 Goals and Constraints

The pedagogical purpose of the WoC-RL exercise is to imagine a version of RL controllers which would be robust to change and capable of adapting their behavior *on the job*. For this reason, WoC-RL makes the idealistic assumption that a controller (hereafter, 'the agent') performs RL endogenously[3] *after deployment*.

The RL procedure operates in accordance with the formulation given in Sect. 3.1. The basic approach can be extended to accommodate additional complexity such as partial observability, stochasticity, and multiple agents. The objective for RL algorithms is to maximize returns, defined as the sum of (discounted) rewards:

$$\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

WoC-RL generalizes the objective of RL by instead seeking to achieve *goals*.[4] This is a more prescriptive notion than *goal-conditioned RL*, an extension of RL in which decisions and value estimates are conditioned on a goal state or embedding thereof (sometimes via universal value function approximation [305]). Goal-conditioned RL is generally used to separate out subtasks and treat them as distinct learning

[3] We leave aside the issue of intrinsic motivation.

[4] Note that WoC-RL is still subject to the limitations described in Claim 4 of Chap. 5.

objectives to hasten the learning of a single complex task [6, 161, 237] or alternatively parameterize a set of related tasks for multitask learning [322, 371]. Regardless, there is no alteration to the reward structure: it is defined a priori and is sampled from every state because while the 'goal' is known, the reward is not.

The goals in WoC-RL effectively replace the notion of reward to become the sole motivation for agent action. Whilst state space regions defined by these goals can indeed be equipped with some associated quantity indicating desirability (which could therefore be said to constitute a 'reward' when reached), the set of state space regions from which this reward can be earned is explicitly specified, thus obviating the need for pointwise sampling of rewards. As such, a WoC-RL agent has access, at any real-valued time t, to the following goal structure $\mathcal{G}$:

$$\mathcal{G}_t = \{\ (S^1, T^1, R^1),\ (S^2, T^2, R^2),\ \dots\ \} \tag{6.1}$$

where each S^i is (a partial specification of)[5] a state, each T^i is a time interval, and each R^i is a positive or negative real. Having access to $\mathcal{G}_t$, the agent knows which (future) states are promised to give a reward or punishment when visited during a specific time interval. If the current state S_t matches a rewarding state (i.e., there exists $(S^i, T^i, R^i) \in \mathcal{G}_t$ such that $S_t \supseteq S^i$ and $t \in T^i$ and $R^i > 0$), a *goal* is said to be achieved; if the current state matches a punishing state ($R^i < 0$), a *constraint* is said to be violated. The relative values of R^i are only useful to an agent in that they aid prioritization between different goals.

The idea is that the tuples in $\mathcal{G}$ will typically persist over long time spans. Nonetheless, in accordance with the requirement to perform work on command, a user can *at any time* modify $\mathcal{G}$ by adding or deleting state space regions: whether goal states in order to 'command' the agent to achieve things, or forbidden regions in order to 'warn' the agent. Assuming that the user is not perfectly wise in specifying the work, it may happen that $\mathcal{G}$ will contain just one or few goals at any time, but also ever more numerous or detailed constraints, as the user's insight increases.

The expected value of the reward of a WoC-RL agent for taking action a in state s can be defined as in traditional RL [331]:

$$r(s, a) \doteq \mathbb{E}[\ R_t \mid S_{t-1} = s,\ A_{t-1} = a\ ] \tag{6.2}$$

but WoC-RL additionally defines the reward signal R as being completely specified by $\mathcal{G}$:

$$R_t \mid \mathcal{G} \doteq \sum[\ R \mid (S, T, R) \in \mathcal{G},\ S_t \supseteq S,\ t \in T\ ] \tag{6.3}$$

where $R_t \mid \mathcal{G}$ reads as 'the reward given the goals'. The long-term *return* as well as the *value function* can again be defined as is traditional in RL [331]; however, neither of these notions serve any purpose in WoC-RL.

The goal structure $\mathcal{G}$ can also be viewed as an explicit description of those states for which a (sparse) reward function would output non-zero values—the only difference

[5] We write $s \supseteq s'$ to mean that s matches s', where s' may be partially specified.

being that this description is explicitly given to the WoC-RL agent without needing to be sampled. Here we see that RL is a restricted case of WoC-RL, namely where a (hypothetical) user would only provide instantaneous goals. That is, RL is obtained if $\mathcal{G}$ is restricted as follows.

$$\forall s, t: \quad \text{either} \quad \mathcal{G}_t = \varnothing \quad \text{or} \quad \exists r \in \mathbb{R} : \mathcal{G}_t = \{(s_t, \{t\}, r)\} \tag{6.4}$$

This stipulates that rewards are immediate and their distribution is not explicitly known a priori. Restricted in this way, WoC-RL turns into traditional RL, as the agent lacks prior information about goals and constraints, never being told ahead of time when or where a reward might be received.

6.2 Planning

WoC-RL changes the optimization problem of RL to that of *planning*: the agent must plan to reach the most valuable state space regions—while avoiding the most punishing ones—as per the goal structure $\mathcal{G}$, while remaining open to changes in $\mathcal{G}$ that can occur at any time.

This framework is inherently *task-agnostic* since the WoC-RL controller must be designed to handle anytime changes in $\mathcal{G}$, whereas RL algorithms typically expect rewards to be stationary. As such, the concept of a policy is not particularly helpful to WoC-RL as it relates to reward. What WoC-RL needs to do is propagate goals through a world model $\mathcal{M}$ in order to generate plans. Such a world model is effectively a learned transition model, ideally of a causal nature, and even more ideally, allowing bidirectional processing. A bidirectional world model allows sensory information to propagate forward as predictions (which, when falsified, can trigger learning), and goals to propagate backward as abductive planning. Nonetheless, even a purely forward world model can be used for deductive planning (disregarding tractability issues), which is easier to express formally, and will be the focus of this section. Abduction will be treated in more detail in Sect. 9.4.

Plan generation can be illustrated by defining an anytime planner:

$$work : Time \rightarrow A$$

that finds the action with the best return according to its current world model $\mathcal{M}$ and goal structure $\mathcal{G}$ (which are retrieved from an implicit storage). Here it is crucial to note that *work is fixed*, i.e., it is not a learnable policy. As prescribed before, $\mathcal{G}$ is changeable by the user but not by the agent. That leaves $\mathcal{M}$—the world model—as *the only adaptable ingredient* of WoC-RL. For pedagogical reasons let us formally specify *work* in terms familiar to the RL community:

$$work(t) \doteq \arg\max_a \; q_*(S_t, a, t \mid \mathcal{G}_t, \mathcal{M}_t) \tag{6.5}$$

where q_* is analogous to the *optimal action-value function* [331], except here it takes not only state s and action a, but also explicitly propagates time t, goals $\mathcal{G}$, and model $\mathcal{M}$:

$$q_*(s, a, t \mid \mathcal{G}, \mathcal{M}) \doteq$$
$$\sum_{s',t'} p_{\mathcal{M}}(s', t' \mid s, a, t)[(R_{t'} \mid \mathcal{G}) + \max_{a'} q_*(s', a', t' \mid \mathcal{G}, \mathcal{M})]$$

where it is assumed the state transition probability p is included in, or can be derived from, model $\mathcal{M}$.[6] It is important to note that the goal structure $\mathcal{G}$ is propagated *unchanged* by the agent even though it can be changed at any time by the user: if that happens at time t'', $work(t'')$ will start using the new goal structure $\mathcal{G}_{t''}$ containing the user's new goal and constraint specifications. This is in line with the *work-on-command* ideal of an agent performing task(s) exactly until the user tells it to do otherwise.

The purpose of the above formula is to demonstrate that WoC-RL is not concerned with sampling rewards or learning a policy. State quality q_* is well-defined in terms of $\mathcal{G}$ and $\mathcal{M}$ which are given to q_*. Action sequences (plans) output by *work* change if and only if $\mathcal{G}$ or $\mathcal{M}$ changes. Thus, here we see how WoC-RL *avoids conflating knowledge and motivation*, which is crucial for the ability to work on command: whenever motivations are (externally) adapted, the latest knowledge should immediately be brought to bear to act toward those new motivations. In contrast, the classical notion of a policy is as a body of (behavioral) knowledge with respect to one target. This offers no provisions for re-applying the same knowledge for new and changeable targets, such as when automating a succession of related business cases.

Still for pedagogical reasons, we sketch below the algorithmic context in which *work* can be embedded. The simplest setup is a loop which also updates the world model $\mathcal{M}$ based on the results of its predictions (gathered in 'pred' below). Note that changes to the goal structure are supposed to occur externally: $work(t)$ always retrieves the actual $\mathcal{G}_t$.

```
pred := ∅;
while true do
    t := WallClock.now();
    expired := { s_t' ∈ pred | t' ≤ t };
    update M_t if 'expired' contains surprises;
    pred := (pred \ expired) ∪ {S_t};
    pred := pred ∪ ⋃{ forward(M_t, s_t') | s_t' ∈ pred };
    a := work(t);
    execute(a);
```

[6] Although in RL it is typically the case that $t' = t + 1$, WoC-RL does not require such discrete time-stepping, assuming instead simply that $t' > t$.

The function *forward* yields zero or more predictions. Without going into the detail of *forward*, we remark that its predictions could carry a likelihood, which may be multiplied for each *forward* step—and *forward* may omit predictions below a certain likelihood threshold. In that case, line 7 could be performed in a loop until 'pred' no longer changes.

We see in this algorithm that $\mathcal{M}_t$ is used for making predictions, causing $\mathcal{M}_t$ to be updated whenever they fail. Now we can also see the naivety of *work* as specified in formula (6.5): if $\mathcal{M}_t$ is used forward for prediction making, then it should also be used backward for abductive planning. Such abductive planning can proceed iteratively, in a similar manner to the prediction making as specified in the above algorithm.

Notice also that the actual *work on command*, as embodied in lines 8 and 9, can be parallelized with the prediction making, since these two lines do not depend on the others, only on the objective time. Thus, we see here how we can finally abolish the 'cognitive cycle', namely the perceive–act–update loop in RL, which is inherited from the sense–think–act loop in GOFAI. However, RL still relies on time-stepping in lockstep with the environment, dangerously assuming that the environment does not change while the agent chooses its next action. The way out is to realize that changes in knowledge and motivation do not have to occur at every time-step or at set times and that plans span over larger time horizons.[7] For as long as predictions succeed, and for as long as the goal structure is not modified by the user, the plans that were or are being formed can be assumed to remain valid—to be best of the agent's knowledge. Thus, a WoC-RL agent could be made to learn, predict, and plan *continually* and *asynchronously* from the environment. An asynchronous open-ended continual inference mechanism will be described later in Sect. 7.3.

Finally, this sketch of WoC-RL illustrates the need for *bootstrapping*: if $\mathcal{M}$ and/or $\mathcal{G}$ would start out empty, no work will be performed. In the early learning phase, the user will need to 'seed' $\mathcal{M}$ with minimal world knowledge, and populate $\mathcal{G}$ with simple goals that can be achieved with that little knowledge. The user may have to act as a 'teacher' or 'mentor', populating $\mathcal{G}$ with progressively more difficult goals and constraints.

Here we have described how a pragmatic perspective on general intelligence calls for the engineering of systems that can perform work on command. WoC-RL illustrates what such a system might look like, and how it would differ from 'vanilla' RL. A crucial aspect is the decoupling of knowledge ($\mathcal{M}$) and motivation ($\mathcal{G}$). Although we have detailed what $\mathcal{G}$ may look like technically, we have so far not delved into the learnable $\mathcal{M}$. To attain general machine intelligence, both $\mathcal{G}$ and $\mathcal{M}$ must be framed in a different manner than has been customary throughout the history of AI, from GOFAI to reinforcement learning. The proposed approach is described in the following chapters.

[7] This would require changing the signature of the anytime planner *work* to return a set of timed actions instead of a single one.

6.3 Anytime Operation

We are concerned with the requirements of deliberative intelligence interacting in a society of asynchronous actors (whether organic or synthetic) to achieve multiple goals—temporally overlapping, potentially non-stationary—at arbitrarily long time horizons (Fig. 6.1). We have determined that such agents must plan over world models, and this raises the following issue. If the worst-case execution time (WCET) of consulting a world model exceeds a certain threshold,[8] the system becomes too unresponsive to be effective. This is exacerbated by the fact that, in the setting of open-ended and lifelong learning, world models inevitably grow in size and complexity.

A general solution to this issue is to enforce *granularity* as a property of the world model. Rather than treating the world state as monolithic, a system should be able to devote computation time to consult only the parts of the world model that are relevant to its goals. However, we argue that granularity alone is not enough with regards to world modeling. Even if the WCET of planning is adequate, there is one final necessary conceptual change. From the perspective of RL, inference is paced by the wall clock of the environment. Mathematically, this may not make a difference when compared to the alternative of an internal, asynchronous clock, but again, the context of real-world situated intelligence changes this. Whereas the WCET of RL inference is negligible compared to the time scale at which the environment evolves, the inference WCET of a deliberative agent is potentially subject to high variability, precluding pacing inferences by the wall clock. Instead, inferences must be computed in *anticipation* of the unfolding of world events, and this requires an internal asynchronous clock: to comply to 'anytime' requirements means to coincide asynchronous internal deliberations with world-synchronous goals. Just as there is

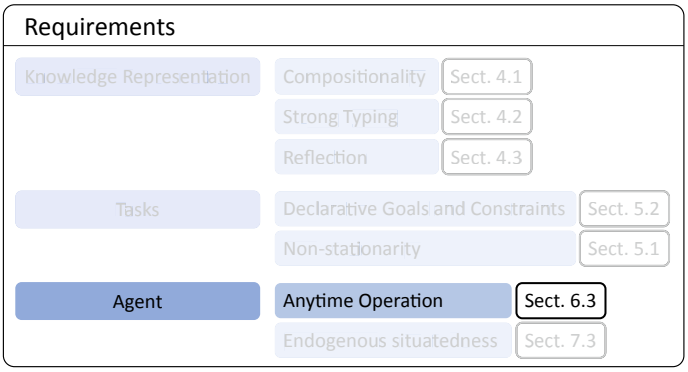

Fig. 6.1 Anytime operation is required to deliver relevant correct action plans on time and asynchronously (relative to system activity). It enables the economically desirable capability of *work on command*

[8] Given a goal, at time t, with a deadline d, the WCET of predicting and planning must be less than $d - t$.

the study of *bounded* rationality [362], there is work on the corresponding concept of *anytime bounded* rationality [27, 150, 241], and we see that this combination is an apt way to unify many of the desirable properties of general intelligence. Unfortunately, common practice in RL is fundamentally incompatible with anytime rationality since it relies both on synchronous coupling with the environment and the concept of 'behavior as a curried planner'. As alluded to previously, anytime operation has not been considered in most demonstrations of RL to date, but we believe that it will become a stubborn hindrance for situated control and multi-agent scenarios.