Chapter 8
Architecture

ACHILLES: The profession of Anteater would seem to be synonymous with being an expert on ant colonies.
ANTEATER: I beg your pardon. 'Anteater' is not my profession; it is my species. By profession, I am a colony surgeon. I specialize in correcting nervous disorders of the colony by the technique of surgical removal.

Hofstadter, 'Gödel, Escher, Bach' [147]

Machine learning excels at inducing mappings from data, but struggles to induce causal hierarchies. In contrast, symbolic reasoning (in particular, when considered as an *expression language*) can represent any form of domain knowledge and can index into code or data via pattern matching.[1] Evidently, reasoning and learning must be robust to both the variability of inputs and the reliability of prior knowledge, learned or imparted. In that regard, Marcus has argued extensively for neuro-symbolic hybrids [210, 213, 214], advocating the complementarity of distributed representations ('neuro') and qualitative localist causal knowledge (symbolic); see also d'Avila Garcez [59]. We explain in this chapter how SCL defines a framework with equivalent goals: although not explicitly 'neural' in operation, the dynamic attention mechanism can be considered to play an analogous role to that of neural connection weights, although perhaps a better guiding metaphor than homogeneous neurons is the 'structural stigmergy' [179] of ant colonies [70, 146]. We then proceed to present a reference system architecture for SCL: its purpose is of course to show how SCL can be realized *in silico* and, by design, to provide sufficient guarantees for its claims (open-ended learning, anytime operation, grounded reasoning, etc.) at the operational level.

[1] Of course, heuristic synthesis of symbolic expressions (e.g. as in Genetic Programming [180]) has been practiced for decades, but has never really been considered as 'mainstream' ML.

J. Swan et al., *The Road to General Intelligence*,
Studies in Computational Intelligence 1049,
https://doi.org/10.1007/978-3-031-08020-3_8

8.1 SCL as a Distributed/Localist Hybrid

Numerous authors have indeed attempted the integration of distributed and localist representations for more than twenty years. However, depending on the original focus of inquiry (reasoning, control theory, ML, or other), 'integration' can serve a rather broad variety of purposes. We now list a few exemplars of these to give some perspective to the purpose of hybridization in SCL; a broader survey can be found in Bharadhwaj et al. [23].

The AKIRA hybrid architecture addresses control rather than learning [257] and is designed around concurrent code fragments. These fragments are organized in a weighted dynamic network of dependencies, competing for budget-constrained computing power on the basis of their expected outcome (goal satisfaction). Focusing on learning instead of control, the DUAL/AMBR architecture [177, 178] controlled symbolic reasoning via spreading activation based on numeric truth values across a network of causal and structural relations, in a manner reminiscent of Copycat [146]. Clarion, another hybrid approach [327, 328], layered symbolic reasoning on top of neural networks, with the aim of enabling top-down and bottom-up learning processes. These were ultimately based on RL, thus imposing, architecture-wide, the fundamental limitations described in previous chapters. In pursuance of another objective, the Sigma architecture attempts to define intelligence in a principled manner [295]: the authors claim 'grand unification', and 'generic cognition' based on graphical models. It is indeed not impossible to think that such a computational substrate, pending further significant work, might become the lingua franca of representation processes able to transcend levels of abstraction. However, the authors have so far limited themselves to the reimplementation of established concepts such as episodic memory [297], RL [296], and a 'standard' model of cognition based on the 'sense–think–act' loop [298].

More recently, the majority of ongoing research on hybrids is unsurprisingly ML-centric, attempting to remedy some of the inadequacies of deep neural networks for reasoning applications. For example, some recurrent neural networks such as LSTM, although designed for sequential/temporal inference, face difficulties with learning long-term dependencies, mainly because their memory consists essentially of compressed (and degradable) input sequences. The Differentiable Neural Computer (DNC) [124] alleviates this problem by coupling the network to an external symbolic memory (other work uses different types of memory augmentation [7, 166, 260]), but does so at the cost of magnifying other shortcomings of DL: the DNC is notoriously harder to train than LSTM (sample inefficiency), and its applicability to arbitrary tasks appears to depend even more than before on the architecture employed and its initial choice of parameters (brittleness). Several improvements have since been proposed [55, 92, 248] and have addressed some of the performance issues but less so the issue of applicability. As of today, the DNC performs reasonably well on latching, copying, associative recall, and simple state machines; the general framework of differentiable computing is preserved but the capabilities of the DNC remain very remote compared to the requirements for general reasoning (abstracting,

analogy making, planning, etc.). Other approaches have made the opposite trade-off, namely that of sacrificing end-to-end differentiability to accommodate more powerful reasoning substrates. They proceed by inserting hand-crafted concepts directly into the RL framework in various forms, e.g. conceptual graphs [152], algorithmic primitives [138, 193], Drescher-style schemata [71, 164], or ontologies and associated auto-encoders [105]. In all of these cases, the agent learns a concept's extension and operational semantics, thus enabling planning via various forms of probabilistic inference. However, as recent research shows, hand-crafting of explicit knowledge structures can be entirely avoided in some cases (so far, mostly problems that can be addressed by direct visual attention). Notably, the PrediNet neural architecture [311], when subjected to an appropriate curriculum, is capable of acquiring some representations with explicit relational and propositional structure. With PrediNet, the neural substrate is used only for learning and does not commit to any specific form of knowledge exploitation: this can be carried out either symbolically (e.g. using predicate calculus, temporal logic, etc.) or, more speculatively, via differentiable 'reasoning models' [69, 225, 294].

These attempts at 'getting the best of symbolic AI and ML' proceed quite literally, essentially reimplementing reasoning (and the necessary supporting representations) in the terms and components of machine learning. We proceed differently. Three aspects of SCL are of particular relevance to what is expected from the reconciliation of ML and symbolic AI:

- Strong typing.
- Fine-grained, open-ended, continual, and compositional inference.
- Emergent resource-aware and goal-directed attention.

We claim that these principles combine the strengths of both approaches: whilst SCL can be provided with prior domain knowledge in any desired form, it is not subject to the problems which plagued GOFAI, since the ability to reflectively reason at the type level allows the *sustained* and *progressive* learning of invariants from the environment. In SCL, learning and planning do not need to be reconciled. By design, both learning and planning are side-effects of the same reasoning process which unfolds in response to the pressure of external goals and constraints over limited knowledge and resources (e.g. inputs, time, computational power and memory, energy, physical agency, etc.). The research cited above pursues the objective of end-to-end representability and actionability of structured knowledge; to this, we add the (orthogonal) requirement of end-to-end controllability, i.e., self-referential goal-directed resource allocation. In the SCL framework, the duality between distributed and local does not concern the representation of functional knowledge (world models, goals, etc. are already unified), but rather the representation of cognition itself. For reasons explained below, we use distributed representations for controlling the reasoning process, itself explicitly represented in the corpus of world knowledge to describe the state of the 'system-in-the-world' (the 'endogenous situatedness' of **Prop-3** in Sect. 7.3); Wang's Non-Axiomatic Reasoning System (NARS) [358] follows a similar approach.

For open-ended learning, inputs may originate from outside the distribution from which knowledge was initially learned. Yet *progress* must nonetheless be sustained, both in terms of continual learning and action. 'Falling outside the system's comfort zone' is not sufficient reason for invalidating acquired knowledge. This would amount to 'stop, erase, and re-learn', similar to the RL 'learn-then-deploy' procedure that, as we have seen, runs counter the desired property of anytime operation. In fact, keeping going might just be the right thing to do: in case foreign inputs happen to be in a semantic continuum with prior distributions, extrapolation would be correct, the system's activity would carry on while vindicating its knowledge over increasingly broad scopes. Of course, in case of a semantic discontinuity, prior knowledge would produce faulty extrapolations and the system might fail to meet its goals. In that respect, the system has to perform two activities, continually and on a case-by-case basis (possibly concurrently). The first consists of extending an initial distribution with foreign inputs and vindicating the current related knowledge. The second is to learn new knowledge from a seemingly novel distribution—the system must also surmise explicit constraints on the relevance of the initial one to guard it against further unwarranted use. This can be achieved by assessing the *degree* to which the learned patterns match the new inputs and propagating the discrepancies across the possible consequences at higher levels of abstraction in the knowledge hierarchy—for these are 'battle hardened' oracles: by construction, they are broader, more reliable and, critically, change less frequently than the lower layers of the hierarchy.

SCL accommodates control heuristics for which truth values are not axiomatic but instead are assessed up to a certain degree (certainty is asymptotic, a tenet we share with Wang [356]). In this approach, truth values are not static, they unfold over time: they are multiplied at each step of inferencing and are also updated whenever new (counter-)evidences become known [241]. At the conceptual level, dynamic numeric truth values thus reflect, quantitatively, inference composition. Interpreted, at the operational level, as the reliability of inference chains, truth values allow to compute the worthiness of resource expenditure (i.e., for further ramifying inference chains) with regards to the goals at hand. This forms the substrate from which goal-directed attention emerges to control the reasoning process, as we will see in the next section.

8.2 Reference Architecture

The ultimate purpose of SCL is to achieve end-to-end consistency of the feedback loops.[2] This is enabled by *endogenous situatedness* as per **Prop-3** of Sect. 7.3 which (1) preserves, anytime, both relevance and correctness, despite the cognitive load possibly overwhelming the inevitably limited computational resources, while (2)

[2] Feedback loops are manifested both at the macro-scale, coupling system and environment, and at the micro-scale (i.e. intra-system), coupling inference chains—formally, the *composition of lenses*, defined in Sect. 9.4.

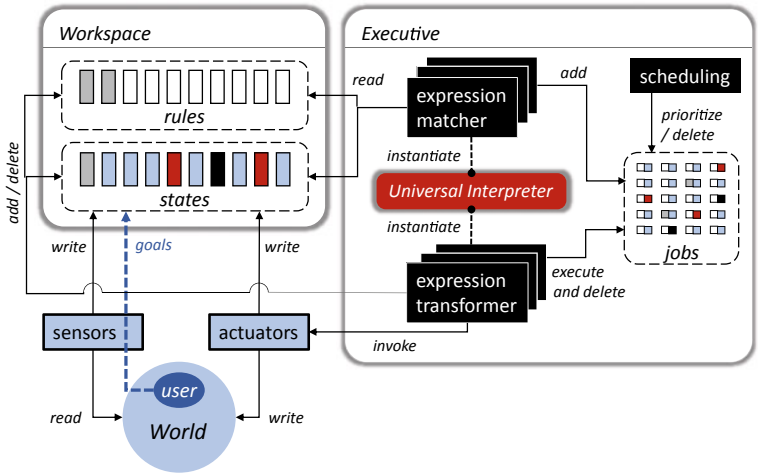

Fig. 8.1 SCL reference system architecture for end-to-end semantic closure. The executive orchestrates a number of asynchronous processes (in black) and provides interfaces to external sensing/actuation data streams (also asynchronous). Scheduling is a resource-aware goal-directed *emergent process*. Rules and states in grey denote axioms. The picture does not distinguish between state modalities—past, present, assumed, predicted, or desired. States in blue pertain to the world, states in red reflect the reasoning process, and states of the executive (memory usage, energy consumption, etc.) are represented in black. The write operation performed by actuators on the workspace carry efference copies; see text for details

deriving such adaptive behavior from the goals at hand, in light of (3) explicit representations of the physical and cognitive state of the system.

Figure 8.1 gives an overview of a reference system architecture. The architecture consists of two main subsystems: a workspace and an executive. The *workspace* contains learned relational knowledge (implemented by rules) and a representation of both world and system states—be they actual (past or present), hypothetical (assumed or predicted), or desired (goals and constraints). Regarding 'relational knowledge', we move away from the vocabulary of causality, which is framed in terms of elementary changes to the 'wiring diagram' of hypothesis chains, as per Pearl's 'do operator'. Instead, in common with the emerging discipline of behavioural control [365] (see Sect. 11.2.3), we adopt the *relational* perspective, which is concerned with the invariants propagated by a relation, as per the notion of *contract* [245]. This offers a more prescriptive framework that is therefore better suited to describing state spaces, particularly those where rules are partially instantiated (e.g. input terms contain wildcards or constraints)—see Sect. 10.1.

The consistency of the workspace is maintained by the *executive* which, besides providing interfaces to sensors and actuators, consists essentially of a scheduler and multiple instances of a universal interpreter. In broad terms, the architecture can be considered a massive fine-grained *production system* [288, 300], in which huge numbers of rules fire concurrently and asynchronously. Some small fraction of the

workspace rules may be provided a priori while the vast majority are produced and maintained dynamically. In addition to rules, the workspace contains states—axioms are also accommodated. Each inferred state has an associated time horizon and an estimate of its likelihood. States qualify both the world, the deliberations of the system ('trains of thought'), and the system itself *embodied* in the world (manifested as memory expenditure, performance profile, physical agency and integrity, etc.).

Sensors and actuators constitute the physical interface of the system to the world, and as such, must be amenable to modeling. For this reason, actuators write *efference copies* in the workspace. An efference copy is an explicit state which encodes the action that was actually executed (by the 'body') in response of the request (by the 'mind') for a desired action: the error between the two allows the system to model the contingencies of its embodiment, i.e., capabilities, limitations and costs (including response latency, energy expenditure, wear and tear, etc.) in a feedback loop generalizing the biologically plausible model of motor control proposed by Wolpert [136]. Errors are useful to learn 'body models' at the edge—'inside-out'—, but not only: there is an inherent dual (and equally important) way to make sense of errors. When body models are well established (i.e., vindicated by significant experience), errors change their meaning: they then signify hidden world states. To take the example used by Wolpert, assuming a carton of milk is full, one will exert a rather strong torque on one's limbs to lift it, only to overshoot the intended end location in case the carton turns out to be empty. In other words, a misaligned efference copy defeats an invalid assumption (a case of abduction). The symmetric feedback loop (i.e. the one pertaining to sensors) is similar, the main distinction being that the inherent duality of errors is operationally *mirrored*. Sensing errors are signals for learning world models at the edge ('outside-in') in response to the invalidation of predictions (a case of deduction). Conversely, when the stationarity of world models is well established (i.e. the reliability of world models is predicted reliably, vindicated by significant experience) sensing errors also change their meaning: they then signify the failure of the sensing apparatus.

Each rule acts as *match-and-transform* schema for typed expressions, encoding patterns that denote subregions S_1, S_2 of a state space S, together with a transfer function $\tau : S_1 \to S_2$. The state space is dynamic: additional dimensions may be synthesized (e.g. via abstraction) at any time by some rules and subsequently also be available for matching and transformation; least recently significant dimensions may conversely be deleted. *Matching* binds values (states, rules or transformation events) to rules, whereas *transforming* combines several such bindings to produce new values (along with a transformation event as a side effect). The allocation of resources for transforming available bindings is prioritized by the scheduler; the most promising bindings will receive computational attention first.

Matching and transformation of SCL expressions is performed via instantiations of a *universal interpreter*. Paths through the state space are composed via the application of interpreter instances and compound SCL inference methods (as introduced in Chap. 7 and discussed in further detail in Sects. 9.2–9.4) to produce inferences (deductive, abductive, or analogical) and state-space dimensions. This interpretation is explicitly compositional and imposes a denotational semantics on expressions.

The unfolding operation of universal interpreter application bears witness to the system's internal state of deliberation: such 'trains of thought' are considered first-class citizens and are duly recorded in the workspace.

The learned ruleset constitutes a fine-grained representation of the entire transition function of the world model. In contrast, axiomatic rules have a different purpose: to implement heuristics for rule and type construction and maintenance. For example, some rules seek to "carve Nature at its joints" [127], by identifying 'surprising' world states such as the failure of a prediction or the unpredicted achievement of a goal. In either case, corrective rule synthesis is applied to the ruleset: this can be considered to impose learned pragmatics on the default denotational semantics. Conversely, the *absence* of surprise vindicates parts of the world model and the rules that implement them: these rules will be deemed more reliable than others that see repeated failures. For completeness, other axiomatic rules include seeking out candidate patterns as a basis for constructing abstractions and analogies, as well as identifying opportunities for generating improved hypotheses. Rule synthesis is addressed extensively in Chap. 10.

Computing resources are inevitably limited and a system must allocate these wisely to cater to arbitrary influxes of inputs and goals. The reference architecture decouples matching from transformation and treats them accordingly as two distinct classes of processes. Matching processes produces *requests* for transformation, termed 'jobs', each of these being associated with a quantitative estimate of its utility with respect to all goals at hand, system-wide. This estimate is based on three main factors: (1) the likelihood of the matched value, which is a function of the reliability of the (chain of) rules that produced it; (2) the reliability of the matching rule; and (3) the relative importance of the goals that may be served (or opposed) by the result of the job. This is revised continually as premises, rules, and goals vary not only in quality (the geometry of the subregions they occupy in the state space) but also quantitatively as per their reliability and relevance. Estimates of utility are primarily used for prioritizing jobs for execution, i.e. in fine, to schedule the execution of transformations competing for resources.

The role of the scheduler in the SCL architecture is merely to re-order jobs continually according to their priorities. For example, previously unattended jobs may jump ahead of new ones as their prospective benefits become apparent, whereas hopelessly unpromising jobs are eventually deleted. This particular scheduling design confers three essential benefits. First of all, it *avoids the scalability issues* inherent in standard job-scheduling theory. The scheduler does not perform explicit allocation but instead slows down relatively less beneficial inference chains, exploiting the fine granularity thereof to its full extent. In the standard acceptance of the term, this is hardly a scheduler at all: SCL scheduling is indeed more a distributed emergent *process* than it is an algorithm. Second, this particular design enforces a fundamental property of the system architecture, namely *endogenous situatedness*. It achieves this by imposing a semantically-grounded control over the execution of the expression transformers: control is based on up-to-date predictions of functional costs and benefits, and balances the cognitive load for the best use of time (deadlines) and available (possibly varying) computing power. Last but not least, semantically-grounded con-

trol implements an *implicit attentional process*: centralized attention algorithms are notorious not only for hindering scalability but also for opposing anytime responsiveness. Instead, an SCL system keeps the cost of attention constant by amortizing the computation of scheduling priorities over the continual transformations of expressions.

The embodiment of the system also imposes limits on the size of its workspace. Although the reference architecture does not impose any specific heuristics to keep the growth of the workspace in check, we mention here a few possibilities. A rule can be evicted on the basis of its reliability: a general trend downwards indicates irrelevance and warrants deletion. There are also other reasons why an otherwise reliable rule can become irrelevant, for example, a temporary change of the stationary regime of the environment or a change in the agent's mission. In such cases, deletion should be avoided (the system may need to revert to previous conditions) and the incriminated rule shall instead be swapped out to some larger auxiliary storage at the cost of incurring extra latency upon its recall (swap in). Whenever a chain of abduction halts before reaching its goal a swap-in request is issued, taking the missing rule signature as its argument. In case no rule is returned, then the system would have found a 'gap' in the world model structure and trigger learning. Inferred states, as we have seen, are qualified by their likelihood; they can be furthermore qualified by their utility: the utility of a prediction increases when its target state matches that of a goal and vice-versa. Low values for both the likelihood and utility can meaningfully trigger the eviction of an inference. Finally, possible heuristics for evicting observed states from the workspace include LRU (least recently used), age, and number of references.