

Online Learning in Perception-Action Systems

Michael Felsberg*, Affan Shaukat, and David Windridge

Dept. of Electrical Engineering, Linköping University,
S-58183 Linköping, Sweden.

`mfe@isy.liu.se`

Centre for Vision, Speech and Signal Processing, FEPS,
University of Surrey, Guildford, UK.

`{a.shaukat,d.windridge}@surrey.ac.uk`

Abstract. In this position paper, we seek to extend the layered perception-action paradigm for on-line learning such that it includes an explicit symbolic processing capability. By incorporating symbolic processing at the apex of the perception action hierarchy in this way, we ensure that abstract symbol manipulation is fully grounded, without the necessity of specifying an explicit representational framework. In order to carry out this novel interfacing between symbolic and sub-symbolic processing, it is necessary to embed fuzzy first-order logic theorem proving within a variational framework. The online learning resulting from the corresponding Euler-Lagrange equations establishes an extended adaptability compared to the standard subsumption architecture. We discuss an application of this approach within the field of advanced driver assistance systems, demonstrating that a closed-form solution to the Euler Lagrange optimization problem is obtainable for simple cases.

Key words: Perception-Action systems, Artificial Cognitive systems, Fuzzy logics, Euler Lagrange optimization, First order theorem proving.

1 Introduction: Perception-Action Systems

In this paper we propose a new approach for designing layered perception-action (PA) systems incorporating a logic-based symbolic processing component. Layered PA systems represent a state-of-the art extension of the standard subsumption architecture [1]. Whereas the latter has generally been focused on fixed-function finite state machines, layered PA systems are more appropriate to adaptive and embodied systems, for which a fixed representational framework would present a serious constraint on learning. Recently, this concept has been demonstrated to produce continuous learning by exploration and adaptation [2].

Thus, by employing the action-precedes-perception paradigm [3] it is possible to entirely bypass the issue of representation; in its place is the issue of the physical grounding of symbols (a symbol in this context being an abstraction of the low-level perception-action coupling through the progressive levels of the

* Corresponding author: Michael Felsberg, URL: <http://www.isy.liu.se/~mfe/>.

subsumption hierarchy until it can be manipulated by an appropriate system (e.g. first order logic). Employing concepts defined in [4], the integration of first order logic within a PA hierarchy was successfully demonstrated in [5]. However, this system employed PROGOL for first order logic induction with all of its attendant instability in a stochastic environment. In the present paper, we set out an approach for integrating a fuzzy logic-based first order theorem prover within a variational perception action framework (specifically an Euler-Lagrange framework) in order to overcome these issues of stability. We further demonstrate that a closed form solution is possible in simple cases.

1.1 States and Mappings in Single Layered System

One difficulty in designing cognitive vision systems is the ambivalent terminology. Therefore, we start by defining a number of terms, in particular different types of *states*.

Virtual system states are states of the system, which do not have a physical realization – except for their physical implementation in terms of memory cells – e.g. variables. These states are called hidden states, internal to the estimator in [4]. *Physical system states* are states of the system, which have a physical realization, e.g. angles of a robotic arm. These states are called visible states, internal to the world in [4].

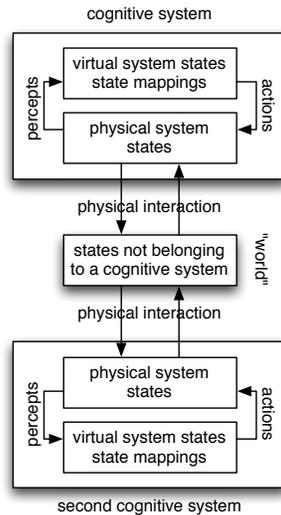


Fig. 1. Illustration of cognitive system(s) and terminology used in this paper.

Percepts are virtual representations of observable physical system states. These states are called visible states, external in [4]. *Actions* are changes of physical system states initiated by virtual states. *State mappings* are mappings taking percepts and virtual system states as input and delivering virtual system states and actions as output. These mappings are called inverse mappings in [4].

A *cognitive system* is a system consisting of all previous items, where at least a part of the mappings is learnable and adaptable. *Physical states not belonging to any cognitive system* are states of the non-cognitive part of the world, or, what we assume as such, e.g. mechanical devices.

Note that percepts and actions are always with respect to system states, i.e., the interface between physical entities and virtual entities lies within the system. The system interacts with the world through its physical states, i.e., the interface to the world is realized by physical interaction of states, e.g., mechanical laws or transmission of light. All terms and connections are illustrated in figure 1.

1.2 Learning in Single Layered System

The paradigm of action-precedes-perception learning [3] is based on the following ratio. The aim is to establish mappings from percepts to actions. For this purpose, both spaces need to be structured.

For the percept space, we know that it has an *extremely high-dimensional extrinsic dimension*. Images of size $M \times N$ lie in a vector space $\mathbb{R}^{M \times N}$. We know further that natural images form a manifold of significantly smaller dimension than their extrinsic dimension, but still of *high-dimensional intrinsic dimension*. The topology of the manifold of natural images is highly complex, including *discontinuities*, wormholes etc. caused by e.g. occlusions.

Hence, learning percept space from percepts only is a very hard problem, including many local minima and problems of convergence. The high dimensionality requires lots of data for learning and the discontinuities make simple methods like gradient descent fail in general.

For the action space, we know that it has a *low-dimensional extrinsic dimension*. Action space usually covers few degrees of freedom. Systems with more than a few 10 dimensions seldomly occur and more than 100 dimensions are hardly ever seen. Actions are performed in a sequence, which has an *intrinsic dimension of 1* and is parametrized by time. The action manifold is highly folded, but *continuous* due to its connection to the physical layer.

Hence, learning action space is a feasible problem and consequently, it is easier to learn percept-actions mappings backward, i.e., by starting from the action side. This can be done in three different ways. In *cognitive bootstrapping* we face entirely unknown mappings and establish them by performing random actions, observing the corresponding percepts, and adding associations for appropriate percept-action pairs. This can only be done as offline learning. During *exploration* we face incomplete mappings and add new mappings by performing random actions and continue as in the bootstrapping case. This corresponds to switching between "working" and "learning" and thus, this is a type of interleaved learning. Finally, and central to this paper, in *adaptation* we face non-perfect mappings and improve them by varying actions and update existing associations. This should happen continuously and corresponds to real online learning.

2 A Layered PA System

In this section we extend the learning-based PA system structure as suggested in [2] to several layers. This extension is required as adaptation requires top-down control. The presented ideas show large overlap with the structure presented in [6] with the main difference that the cited work only considers perception.

2.1 States and Mappings in Multi-Layer System

The major change in terminology compared to the previous section is the extension of percepts and actions to higher levels. Furthermore, the terms control and feedback are introduced. *Percepts* are virtual representations at level n of observable system states at level $n - 1$ if $n > 0$ or virtual representations of observable physical system states at level 0. *Actions* are changes of system states at level $n - 1$ initiated by virtual states at level n if $n > 0$ or physical system states initiated by virtual states at level 0. *Feedback* is the output of level n that is considered as percepts at level $n + 1$ and *control* is the input to level n that is considered as actions at level $n + 1$.

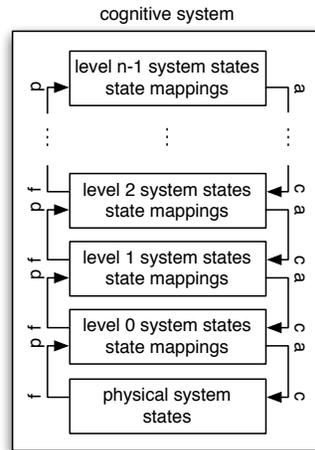


Fig. 2. Layered system architecture and related terminology: percepts (p), actions (a), feedback (f), and control (c).

These concepts allow us to build a system with several layers of virtual states and mappings, as illustrated in figure 2. Note that level 0 can be considered as an innate (engineered) level and n grows with time. Note also the similarity to the subsumption architecture [1] with the major difference that the latter is using finite state machines.

2.2 Learning in Multi-Layered System

The considered system does not have distinct modes for learning and working. From the beginning, it has no capabilities except for level 0. This level can be

considered as a (virtual representation of) physical state predictor. Whenever the prediction fails, a feedback is triggered to level 1, indicating a need for changing the control of level 0, i.e., requesting an action at level 1. In this way, level 1 is embedded in a percept-action cycle and acts as a predictor for level 0 states. This leads to bootstrapping of level 1, using the mechanisms described earlier. When the level 1 predictions start to work, bootstrapping of level 1 can be considered completed.

The feedback of level 1 (being sent to level 2), becomes stronger with time and eventually triggers a similar bootstrapping process at level 2. Meanwhile, level 1 still adapts, continuously trying to improve its predictions. By sending different control signals from level 2 to level 1, level 1 develops different modes of prediction, i.e., it starts to make different predictions based on the top-down control. If the feedback from level 0 (the percept of level 1) becomes stronger than a triggering threshold, level 1 will generate a new mode of prediction, thus exploratory learning is applied as described earlier.

Repeating this procedure through more and more levels lead to a system with increasingly better capabilities w.r.t. solving a task of appropriate complexity. It is important to choose the complexity such that exactly one new level needs to be generated in each step, otherwise learning might become slow or entirely break down, very similar to learning in humans.

2.3 Adaption of a Layer

The central part of online learning is the adaptation process performed by a single layer m . Thus, we restrict our system diagram to a single layer in figure 2. Note the similarity to PAC modules [7] with the difference that in PAC modules the feedback is part of the action output, i.e., the percept at level m was given by a part of the action at level $m - 1$, instead of the feedback from level $m - 1$. We call the extended PACs PACFs in what follows.

Similar to the PACs, the PACFs are related to the Neisser perceptual cycle [8] and can be interpreted as an abstract tracking process that is looping through the reflexive transitive closure of (state – system model – prediction – matches – observation – update). The matching of predictions and observations (percepts) defines the internal modelling error and drives adaptation and feedback. Small errors lead to adaptation, whereas large errors lead to feedback and exploration or bootstrapping at the next higher level.

Let us define: p_t is the percept at time t (I in [4]), r_t is the internal state at time t , a_t is the action at time t , c_t is the control at time t , f_t is the feedback at time t , and \tilde{p}_t is the expected percept a time t . Furthermore, we define two mappings: v is the internal state update

$$r_{t+1} = v(c_t, \rho(\tilde{p}_t - p_t), r_t) , \tag{1}$$

where ρ is some error function. u is the prediction generation

$$\tilde{p}_t = u(r_t) , \tag{2}$$

and the action and feedback is given by (a subset of) \tilde{p}_t and $r(t)$, respectively.

The feedback, i.e., $r(t)$, influences the state vector r_{t+1} and leads to internal adaption through (1). Furthermore, it might lead to changes in the control signal for a future time step (c_{t+t_c}). Actions might lead to new percepts with unknown delay p_{t+t_p} . A scheme for two interacting PACFs is illustrated in figure 3.

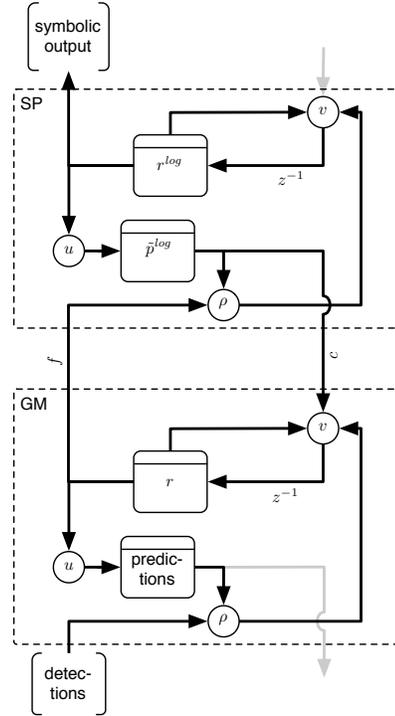


Fig. 3. State diagram of two interacting PACFs. Note that in the upper part p^{log} corresponds to f . In the lower part, p corresponds to detections and \tilde{p} to predicted detections. z^{-1} denotes a unit time delay.

The goal of each layer is to achieve a low prediction error $\rho(\tilde{p}_t - p_t)$ and at the same time to update the internal state as little as possible. The first criterion is obvious: if the prediction error is low, the system provides good predictions of percepts caused by its actions, i.e., the system shows a high degree of *flexibility* to model percept-action loops. The second point might be less obvious, but has partly been discussed in [4] and can be further motivated by minimizing efforts to change the system and by implicitly requiring the control to be smooth.

One implication of the action-precedes-perception paradigm is that actions are continuous for consistent trajectories through percept space. However, percept space is discontinuous and therefore it is easier to measure the consistency of percept sequences by continuity of actions. Therefore, we consider fewer changes

of the control signals as an indicator for correct feedback. This corresponds to the *generalization capability* of the layer under consideration.

Hence, we obtain a classical generalization versus flexibility tradeoff, a typical bias-variance dilemma.

2.4 An Implementation of Learning

One possibility to formulate the learning problem that we face is in terms of an objective function that is to be minimized. Obviously, we would like to have a good prediction independently of what the control c_t and the percept p_t are (fidelity). Furthermore, small changes in the percepts should result in small changes of the internal state r . These requirements result in

$$\varepsilon(u, v) = \int \Psi(|\nabla_p r_{t+1}|) + \rho(\tilde{p}_t - p_t) dc_t dp_t , \quad (3)$$

where Ψ is some error norm applied to the gradient of r , e.g. the L1-norm (which will basically count the occurrences of changes of r). In order to optimize this equation, we plug in (1) and (2)

$$\varepsilon(u, v) = \int \Psi(|\nabla_p v(c_t, \rho(u(r_t) - p_t), r_t)|) + \rho(u(r_t) - p_t) dc_t dp_t . \quad (4)$$

The control signal c_t is obtained from the SP (logic) layer as a function of the previous state r_{t-1} (cf. fig. 3): $c_t = L(r_{t-1})$ with the Jacobean $J(r_{t-1}) = \frac{\partial c_t}{\partial r_{t-1}}$. Substituting c_t and $\tilde{v}(r_{t-1}, u(r_t), p_t, r_t) = v(L(r_{t-1}), \rho(u(r_t) - p_t), r_t)$, we obtain

$$\varepsilon(u, \tilde{v}) = \int \{\Psi(|\tilde{v}_p(r_{t-1}, u(r_t), p_t, r_t)|) + \rho(u(r_t) - p_t)\} J(r_{t-1}) dr_{t-1} dp_t . \quad (5)$$

We compute the Euler-Lagrange equations (where we omit the argument of \tilde{v})

$$0 = \varepsilon_u = \{\Psi'(|\tilde{v}_p|) \frac{\tilde{v}_p}{|\tilde{v}_p|} \tilde{v}_{p,u} + \rho'(u(r_t) - p_t)\} J(r_{t-1}) \quad (6)$$

$$0 = \varepsilon_{\tilde{v}} = -\text{div}_p \left\{ \Psi'(|\tilde{v}_p|) \frac{\tilde{v}_p}{|\tilde{v}_p|} \right\} J(r_{t-1}) = -\Psi''(|\tilde{v}_p|) \tilde{v}_{pp} J(r_{t-1}) \quad (7)$$

In order to get the equations for v instead of \tilde{v} , we substitute v back. Note that $\tilde{v}_p = -v_{\rho} \rho'$. Finally, we obtain the mappings u and v by gradient descent:

$$u_{\text{new}} = u_{\text{old}} - \alpha \varepsilon_u \quad (8)$$

$$v_{\text{new}} = v_{\text{old}} - \beta \varepsilon_v \quad (9)$$

for some suitable step-lengths α and β which might also vary with time. Again, terminology is adapted to [4]. If we use a linear model for u giving $u(r_t) = u_r r_t$, the two updates contain (18) and (19) in [4] as special cases.

Looking at the Euler-Lagrange equation (6) and (7) in more detail, we see that the Jacobian of the next higher level directly influences the updates of

the mapping u and v : it might suppress or amplify updates depending on the relevance for the further processing. How this Jacobean is obtained in a logic system will be part of the remainder of the paper.

The influence of internal entities of the subsymbolic part can be understood as follows: The first term in (6) and the term in (7) have a smoothing effect on v and u , whereas the second term in (7) improves fidelity.

3 The Logic Module

We will demonstrate how fuzzy theorem proving within the SP module may be linked to the PACF module immediately below it, thereby demonstrating how symbolic logic can be integrated within the Euler Lagrange framework. In order to do this it is necessary to quantify, in closed-form, the effects of logical resolution with respect to a given query. We do this as follows.

3.1 Fuzzy Theorem Proving

Fuzzy logic is the logic applicable to fuzzy sets, i.e. sets for which there are *degrees of membership*. This is usually formulated in terms of a membership function valued in the real unit interval $[0, 1]$. Various fuzzy logics are possible within this framework; membership functions (and therefore *truth values*) can be single values, intervals or even (most generally) Borel sets within the unit interval $[0, 1]$. [9] argues that fuzzy logic programming is well suited to the implementation of methodologies relating to reasoning with uncertainty.

The propagation of truth values through logic rules is carried by means of an *aggregation operator*. The *aggregation operators* subsumes conjunctive operators (T-norms; min, prod etc) or disjunctive operators (T-conorms; max, sum, etc) and hybrid operators (combination of previous operators) [10].

The concept of aggregation operator also extends to the notion of implication, which is implemented as a residuum of the t-norm. A left-continuous t-norm T (i.e. an intersection of two fuzzy sets say x and y) is generally specified by a binary operation on the unit interval; i.e., it is a function of the form [11],

$$i : [0, 1] \times [0, 1] \Rightarrow [0, 1]$$

thus there is a unique binary operation \Rightarrow on $[0, 1]$ such that,

$$T(\gamma, x) \leq y \quad \text{iff} \quad \gamma \leq (x \Rightarrow y), \quad \forall (x, y, \gamma) \in [0, 1]. \quad (10)$$

The operation above is known as the *residuum* of the t-norm. The residuum of a t-norm T is generally denoted by R . The residuum R of a t-norm T forms a right adjoint $R(x, -)$ to the functor $T(-, x)$ for each x in the lattice $[0, 1]$. In standard *fuzzy logic*, conjunction is generally interpreted by a t-norm T where as the residuum R defines the role of implication. If \Rightarrow defines the residuum of a left-continuous t-norm T , then

$$(x \Rightarrow y) = \sup\{\gamma \mid T(\gamma, x) \leq y\} \quad (11)$$

thus, $\forall(x, y, \gamma) \in [0, 1]$,

$$(x \Rightarrow y) = 1 \quad \text{iff} \quad x \leq y \quad \text{and} \quad (1 \Rightarrow y) = y \quad (12)$$

This allows for the representation of clauses as implication of a head predicate with respect to a conjunct of body literals. Consequently, modus ponens arguments and (more generally) theorem proving via resolution are thus achievable within fuzzy logic.

3.2 Fuzzy Prolog Syntax

We implement our method within Ciao Prolog or Fuzzy Prolog system, which provides a complete library of fuzzy logic semantics amalgamated with complete Prolog system supporting ISO-Prolog [10]. Given that A is an atom then a fuzzy fact can be represented as, $A \leftarrow v$, where as v is a truth value, an element in the Borel Algebra, $\mathcal{B}([0, 1])$, which can be considered as the power set of the sets of continuous subintervals on $[0, 1]$ [12].

If A, X_1, \dots, X_n are atoms,

$$A \leftarrow_F X_1, \dots, X_n \quad (13)$$

represents a *fuzzy clause* where F is an *interval-aggregation* operator of truth values in $\mathcal{B}([0, 1])$, defined as constraints over the domain $[0, 1]$. F here, induces a *union-aggregation* [10]:

$$F'(B_1, \dots, B_n) = \cup\{F(\varepsilon_1, \dots, \varepsilon_n) | \varepsilon_j \in B_j\} \quad (14)$$

A *fuzzy query* is a tuple, if A is an atom, and v is a variable representing a truth value in $\mathcal{B}([0, 1])$, then a fuzzy query is represented as,

$$v \leftarrow A? \quad (15)$$

A *fuzzy program* is a collection of facts and rules (i.e. a subset of the Herbrand base mapped into $\mathcal{B}([0, 1])$ along with a set of clauses).

Meaning is interpreted as the *least model* of a program; the concept of 'least' being defined under set inclusion with regard to subsets of the Herbrand base and under lattice inclusion with respect to fuzzy values within the Borel set $\mathcal{B}([0, 1])$.

This is defined in [10] as follows; if T_P is a one-step logical consequence operator for the fuzzy program, P such that $T_P(I) = I'$ with $I' = \langle B_{I'}, V_{I'} \rangle$, then:

$$\begin{aligned} B_{I'} &= \{A \in B_P | \text{cond}\} \\ V_{I'}(A) &= \cup\{v \in \mathcal{B}([0, 1]) | \text{cond}\} \end{aligned}$$

where:

$\text{cond} = A < -v$ i.e. a ground instance of a fact in P or $A < -_F A_1, A_2, \dots, A_n$

where F is a ground instance of a clause in P

($B_P = \text{Herbrand base of } P$)

The meaning of a fuzzy program, P , is thus its *least fixed point*, $lfp(T_P) = I$ ($T_P(I) = I$), with respect to the allocation of fuzzy values to predicates from its Herbrand base, where I is an *interpretation* i.e. a mapping of truth values to a subset of the Herbrand base.

We thus equate the notion of a least model, I , with stability under the operation of resolution, such that all facts within I are consistent with each other, being thus either axiomatic or provable from axioms. This notion will become important later when we subsume fuzzy Prolog within a hierarchical Euler-Lagrange framework, for which I will constitute a set of internal or *virtual* system state representing the *logical completion* of the sparse fuzzy input 'perceptual' input p . The least model is thus the optimal self-consistent world model on the basis of an Occam's-razor-like closed world assumption.

3.3 Operational Semantics of Fuzzy Prolog

According to [10], the procedural semantics of *Fuzzy Logic* incorporating constraint logic programs can be defined in terms of a virtual interpreter that works as a sequence of transitions between different states of the system. The state of a *transition system* in a computation always maintains a tuple $\langle A, \sigma, S \rangle$, where as A is the current goal, σ is a substitution representing the instantiation of variables and S is a constraint store that holds the truth value of the goal at that specific state [10]. The current goal A contains the literals that the interpreter has to prove, along with the constraints it has to satisfy i.e. S ; the *constraint store* contains all the constraints that have been assumed by the interpreter to this point. Thus a generic transition is a pair of the tuple (goal/constraint store); $\langle A, \sigma, S \rangle \longrightarrow \langle A', \sigma', S' \rangle$. It states the possibility of going from state $\langle A, \sigma, S \rangle$ to state $\langle A', \sigma', S' \rangle$.

The computation comprises a sequence of transitions, which starts with A set as the initial goal, $\sigma = \emptyset$ and S set to true (with the assumption that there are neither previous instantiations nor initial constraints). The sequential transitions halt when the first argument A is empty, where as σ and S represent the final result. The whole transition process can be more formally defined as follows:

Given that $q \leftarrow v$ is a fact of the program P , and r is a component of the goal A , θ is the '*most general unifier*' of q and r , and μ_r is the truth variable for r ,

$$\langle A \cup r, \sigma, S \rangle \longrightarrow \langle A\theta, \sigma \cdot \theta, S \wedge \mu_r = v \rangle \quad (16)$$

if $q \leftarrow_F B$ is a rule of the program P , θ is the '*most general unifier*' of q and r , c is the constraint that defines a truth value obtained via the application of the aggregator F on the truth variables of B ,

$$\langle A \cup r, \sigma, S \rangle \longrightarrow \langle (A \cup B)\theta, \sigma \cdot \theta, S \wedge c \rangle \quad (17)$$

if none of the above cases are applicable then the transition fails,

$$\langle A \cup r, \sigma, S \rangle \longrightarrow \text{fail} \quad (18)$$

We implement *fuzzy theorem proving* within an Euler-Lagrange framework as follows.

3.4 Implementing Fuzzy Theorem Proving with a PACF

The complete success set of goals $p(\hat{x})$ for a program P is denoted $SS(P) = \langle B, V \rangle$ with

$$\begin{aligned} B &= \{p(\hat{x})\sigma \mid \langle p(\hat{x}), \emptyset, true \rangle \rightarrow^* \langle \emptyset, \sigma, S \rangle\} \\ V(p(\hat{x})) &= \bigcup \{v \mid \langle v, \emptyset, true \rangle \rightarrow^* \langle \emptyset, \sigma, S \rangle\} \end{aligned}$$

(\rightarrow^* is the iterated transition sequence from the goal to the empty set).

The declarative and operational semantics of Ciao Prolog can be shown to be equivalent i.e. such that $SSP(P) = lfp(T_P) = I$, where the least fixed point I is also the least model of program P . This means that a self-consistent world model incorporating the sparse perceptual facts and highwaycode/ECOM rules; (the *Extended Control Model* i.e., ECOM, suggests that humans employ a hierarchical perception-action model within the context of driving [13]), can be created via exhaustive querying. We shall use this to generate the logic-level virtual road model at time t , i.e. V_t^{log} , and relate this to a logic-level virtual system state, r_t^{log} used in the PACF.

Thus, $V_t^{log} = lfp(T_P)$ where $P_t = \langle \bigwedge_{t'} \{Is_past(t-t') \wedge p_{t-t'} \times \tilde{f}_{t-t'}\}, R \rangle$, such that R is the clause set embodying the highwaycode and ECOM rules ($\bigwedge_{t'}$ is the union via conjunction over all t' ; $t' > 0$). P is thus an amalgam of the historical perceptual observations and the known *a priori* rules. $\tilde{f}_t \in \{0, 1\}$ is a consistency-based 'inclusion' multiplier that determines whether p_t is included in the historical set of percepts (see later). The program P_t is thus regenerated with each temporal iteration $t = t + 1$. Note that we write p_t as a shorthand for $\langle v' \leftarrow p_t, p_t \rangle$ i.e. we always consider the percepts p_t along with their fuzzy membership allocations.

We may equate r_t^{log} with P_t via reasoning as follows. A fact g may be amalgamated with a program P that already embodies the facts F and rules R so as to create a new program P' via the relation $P' = g \wedge P = \langle g \wedge F, R \rangle$. The logic-level PACF prediction generation, $\tilde{p}_{t+1} = u^{log}(r_t^{log}, p_t)$, can thus be modelled on the logic layer by updating P_t so as to include the current perceptual observations p_t i.e. $P'_t = Is_current(t) \wedge p_t \wedge P_t$.

A query $v \leftarrow p_{t+1}$? directed at the program P'_t determines the \tilde{p}^{log} , i.e. the predicted future perceptual state \tilde{p}_{t+1} . We denote this $P'_t \{ \tilde{p}_{t+1} = v \leftarrow p_{t+1} ? \}$. Hence, the function u^{log} is defined:

$$\tilde{p}_{t+1} = u(r_t^{log}, p_t) \equiv P'_t \{ \tilde{p}_{t+1} = v \leftarrow p_{t+1} \} : P'_t = r_t^{log} \wedge p_t \wedge Is_current(t)$$

Actions a are passed back to the lower PACF in the form of modified confidences on p_t derived from the extrapolation of the globally-consistent model i.e. $a = \tilde{p}_t$.

There is no control specified on the topmost logic layer (if this were to exist it would perhaps function as a clause update or rule induction procedure). Thus we can write $r_t^{log} = v^{log}(f_{t-1}^{log}, r_{t-1}^{log})$. This is thus the temporal update function conditional on the feedback f_{t-1}^{log} i.e. $r_t^{log} = (p_t \times \tilde{f}_t) \wedge Is_past(p_t) \wedge r_{t-1}^{log}$.

The function $\rho(\tilde{p}_t - p_t)$ acts as an internal gating function on perceptual modifications $a = \tilde{p}_t$, so that excessive disparity between prediction and observation $|\tilde{p}_t - p_t| > threshold$ 'turns off' an internal confidence flag regarding the global

consistency of the perceptual p_t , i.e. such that $\tilde{f}_t = 0$ (it is unity otherwise). The action feedback to the lower level is always $\langle v' \leftarrow \tilde{p}_t, \tilde{p}_t \rangle$ (i.e. the predicted percepts plus their confidences) by default, supplemented by this warning flag. $\langle v' \leftarrow \tilde{p}_t, \tilde{p}_t \rangle$ can thus be treated as a 'logical prior' on observations.

Thus in the function: $[f_t^{log} \ a_t^{log}] = w(\rho(\tilde{p}_t - p_t), r_t^{log})$

we have that:

$a_t^{log} = \langle v' \leftarrow \tilde{p}_t, \tilde{p}_t \rangle$ and $f_t^{log} = \tilde{f}_t = H(|\tilde{p}_t - p_t| - threshold)$

with H the Heaviside step function (or Logistic function for analytic continuity, if necessary).

3.5 Integration with Euler Lagrange Optimization

As regards the Euler Lagrange optimization, we may omit consideration of Ψ^{log} (characterizing the cost of changes in r^{log} with respect to changes in p). This is because p is sparse with respect to V^{log} as a whole, as well as being limited to a single temporal slice, and so we would not expect much change in r^{log} (the 'global' perceptual picture) unless p is a critical predicate (assuming that p is globally consistent and not flagged by \tilde{f}). Thus, in general, to a first order of approximation, r^{log} is modified only in terms of the fuzzy truth values referring to p . However, for the cases that will prove quantifiable later, with only a few predicates, the change in r^{log} is disproportionately large due to the lack of constraint on V^{log} .

Ψ^{log} is thus not applicable the logic level; change in r^{log} with respect to p is dictated by the resolution procedure and will tend to stabilize naturally as more context is accrued.

We thus consider only the Jacobean 'interface' $\frac{\partial p^{log}}{\partial a^{log}} \equiv \frac{\partial c}{\partial f}$ at the logical level.

3.6 The Logic-level Jacobean: A worked example

This process can be elaborated with more perspicuity by the following simple example,

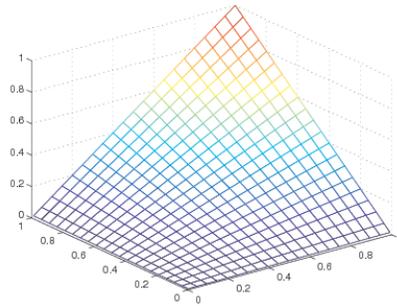


Fig. 4. Graph of the Product T-norm

let the terms pr_1 , pr_2 , and pr_3 represent the predicates; $obj_position_left_road$, $obj_position_driver_road$, and $velocity_leftwards$ respectively, where $t1 = past$ and $t2 = current$;

$$\begin{aligned} pr_1(car_1, t2) &\leftarrow 0.7, pr_2(car_1, t1) \leftarrow 0.9, pr_3(car_1, t2) \leftarrow 0.8; \\ driver_int(turn_left, X) &\leftarrow T_{prod} \quad pr_1(X, t2), pr_2(X, t1), pr_3(X, t2). \end{aligned}$$

The complete Borel set is confined to discrete confidence values defining the truth values for each fuzzy predicate (i.e. feature input) by applying a *max* operator to the input predicates. A simple Product T-norm is used for logical resolution of clauses, $T_{prod}(x, y) = x \cdot y$ (which is an ordinary product of two real numbers). The product T-norm is a strict Archimedean t-norm semantically used for strong *conjunction* in fuzzy logics (refer to figure 4).

Three temporal facts have been provided here, $pr_1(car_1, t2)$, $pr_2(car_1, t1)$, and $pr_3(car_1, t2)$ whose truth values are the unitary intervals $[0.7, 0.7]$, $[0.9, 0.9]$, and $[0.8, 0.8]$, respectively, and a fuzzy clause for the *driver_int* predicate, which is defined by an *aggregation operator* that is a *Product T-norm*. Lets consider a *fuzzy goal*, $\mu \leftarrow driver_int(turn_left, X)$? the first *transition* in the computation would be:

$$\langle \{(driver_int(turn_left, X)), \xi, true\} \rangle \rightarrow \langle \{pr_1(X, t2), pr_2(X, t1), pr_3(X, t2)\}, \xi, \mu = ((\mu_{pr_1} \cdot \mu_{pr_2}) \cdot \mu_{pr_3}) \rangle$$

the goal is unified with the clause and the constraint corresponding to Product T-norm is added. The next *transition* leads to the state:

$$\langle \{pr_3(X, t2)\}, \{pr_2(X, t1)\}, \{X = car_1\}, \mu = ((\mu_{pr_1} \cdot \mu_{pr_2}) \cdot \mu_{pr_3}) \wedge \mu_{pr_1} = 0.7 \rangle$$

after $pr_1(X, t2)$ is unified with $pr_1(car_1, t2)$ and the constraint defining the truth value of the fact is added, the computation follows towards the next iteration:

$$\langle \{pr_3(X, t2)\}, \{X = car_1\}, \mu = ((\mu_{pr_1} \cdot \mu_{pr_2}) \cdot \mu_{pr_3}) \wedge \mu_{pr_1} = 0.7 \wedge \mu_{pr_2} = 0.9 \rangle$$

The computation ends with the following iteration:

$$\langle \{\}, \{X = car_1\}, \mu = ((\mu_{pr_1} \cdot \mu_{pr_2}) \cdot \mu_{pr_3}) \wedge \mu_{pr_1} = 0.7 \wedge \mu_{pr_2} = 0.9 \wedge \mu_{pr_3} = 0.8 \rangle$$

Since $\mu = ((\mu_{pr_1} \cdot \mu_{pr_2}) \cdot \mu_{pr_3}) \wedge \mu_{pr_1} = 0.7 \wedge \mu_{pr_2} = 0.9 \wedge \mu_{pr_3} = 0.8$ entails $\mu \in [0.50, 0.50]$, the answer to the query $driver_int(turn_left, X)$ is $X = car_1$ with its truth value in the unitary interval $[0.50, 0.50]$. Thus given any arbitrary truth values for the fuzzy facts $pr_1(X, T)$, $pr_2(X, T)$, and $pr_3(X, T)$ i.e. $[x1, x1]$, $[x2, x2]$, and $[x3, x3]$, respectively, the truth value for the *driver intention* can be summarized into the following general formula:

$$\mu_{int} = \{T_{prod}(\mu_{pr_1}, \mu_{pr_2}, \mu_{pr_3}) \wedge \mu_{pr_1} = x1 \wedge \mu_{pr_2} = x2 \wedge \mu_{pr_3} = x3\}$$

where as $\mu_{int} \in [0, 1]$. Thus $\mu_{int} = x1 \times x2 \times x3$ in this very simplified case. If we thus treat μ_{int} as the simplified output a^{log} from the logical level with respect to changes in $p = \{\mu_{pr_1}, \mu_{pr_2}, \mu_{pr_3}\}$, then the Jacobean $\frac{\partial p}{\partial a^{log}}$ becomes: $[x1 \times x2 + x1 \times x3 + x2 \times x3]$. Hence as such, we can achieve the closed form solution to a multi-level PACF in terms of a top level jacobean, thereby it is possible to find a solution to a multi-layer Euler-Lagrange problem.

4 Conclusion and Future Work

Having described the multi-layer PACF and its implementation within an Euler-Lagrange framework, we have demonstrated the closed form solution to a *first-order fuzzy* theorem proving. Thus it is possible to build a grounded hierarchical Perception-Action framework with *first-order* symbol processing capability of working in a stochastic environment. The proposed learning structure has been evaluated in a single layer system using offline learning [14]. What remains is to evaluate the proposed framework for online learning in a multi-layer system.

5 Acknowledgements

This research work has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 215078 (DIPLECS) and 247947 (GARNICS).

References

1. Brooks, R.A.: Intelligence without representation. *Artif. Intell.* **47** (1991) 139–159
2. Felsberg, M., Wiklund, J., Granlund, G.: Exploratory learning structures in artificial cognitive systems. *Image and Vision Computing* **27** (2009) 1671–1687
3. Granlund, G.H.: The complexity of vision. *Signal Processing* **74** (1999) 101–126
4. Rao, R.P.N.: An optimal estimation approach to visual perception and learning. *Vision Research* **39** (1999) 1963–1989
5. Windridge, D., Kittler, J.: Perception-action learning as an epistemologically-consistent model for self-updating cognitive representation. *Advances in Experimental Medicine and Biology* **657** (2010)
6. Rao, R.P.N., Ballard, D.H.: Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience* **2** (1999) 79–87
7. Felsberg, M., Granlund, G.: Fusing dynamic percepts and symbols in cognitive systems. In: *International Conference on Cognitive Systems*. (2008)
8. Neisser, U.: *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W. H. Freeman, San Francisco (1976)
9. Shapiro, E.Y.: Logic programs with uncertainties: a tool for implementing rule-based systems. In: *IJCAI'83*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1983) 529–532
10. Vaucheret, C., Guadarrama, S., Mu noz-Hernández, S.: Fuzzy prolog: A simple general implementation using clp(r). In: *Proceedings of the 18th International Conference on Logic Programming*, London, UK, Springer-Verlag (2002) 469
11. Klir, G.J., Yuan, B.: *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. 1st edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (1995)
12. Mu noz-Hernandez, S., Sari Wiguna, W.: Fuzzy cognitive layer in robocupsoccer. In: *IFSA '07*, Berlin, Heidelberg, Springer-Verlag (2007) 635–645
13. Hollnagel, E., Woods, D.D. In: *Joint Cognitive Systems: Foundations of Cognitive Systems Engineering*. CRC Press, FL 33487-2742 (2005) 149–154
14. Felsberg, M., Larsson, F.: Learning Bayesian tracking for motion estimation. In: *International Workshop on Machine Learning for Vision-based Motion Analysis*. (2008)