Multi-Level Knowledge Processing in Cognitive Technical Systems

Thomas Geier and Susanne Biundo

Abstract Companion-Systems are composed of different modules that have to share a single, sound estimate of the current situation. While the long-term decision-making of automated planning requires knowledge about the user's goals, short-term decisions, like choosing among modes of user-interaction, depend on properties such as lighting conditions. In addition to the diverse scopes of the involved models, a large portion of the information required within such a system cannot be directly observed, but has to be inferred from background knowledge and sensory data—sometimes via a cascade of abstraction layers, and often resulting in uncertain predictions. In this contribution, we interpret an existing cognitive technical system under the assumption that it solves a factored, partially observable Markov decision process. Our interpretation heavily draws from the concepts of probabilistic graphical models and hierarchical reinforcement learning, and fosters a view that cleanly separates between inference and decision making. The results are discussed and compared to existing approaches from other application domains.

1 Introduction

Early computers were separated from the physical world they resided in by nearly impervious barriers—punch cards, light bulbs, and later monochrome low resolution screens and keyboards. Experts were required to communicate with those machines using now obsolete (and possibly obscure) machine languages.

Thomas Geier
Institute of Artificial Intelligence, Ulm University
e-mail: thomas.geier@uni-ulm.de

Susanne Biundo
Institute of Artificial Intelligence, Ulm University
e-mail: susanne.biundo@uni-ulm.de

But technology evolves, and today's technical systems have become complex; very distinguished from their ancestors. A current smart phone, easily fitting into a pocket, is connected to its environment through a multitude of sensors and output devices—high-resolution touch screens, video cameras, gyroscopes, heartbeat detectors, permanent connection to the internet, and much more. The technology that is used to process this plethora of information has also advanced significantly. Automatic speech recognition has become viable, and development of touch screen text input has reached a point where organic swipes yield the intended words with high accuracy.

Current technical systems are thus embedded much deeper in their physical environment, and using these devices has become possible for laypersons; it is not requiring any exceptional skills anymore. Still there appears to be a gap between the complexity the average user is able to handle, and the functionality that could be provided by a modern technical device.

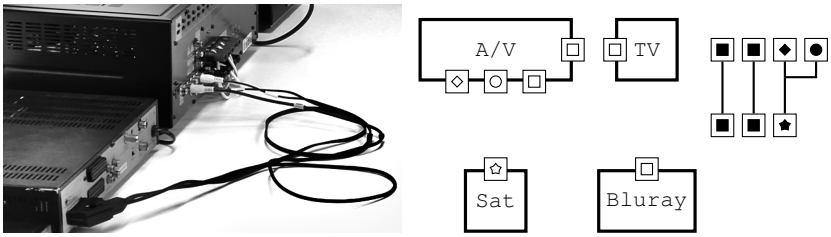
To bridge this gap we have the vision that modern technical systems shall be cognitive. This means that they possess abilities such as attention, memory, reasoning, decision making and comprehension, that let them exploit their deep embedding into the physical world to offer their functionality in a more accessible way to human users.

In this chapter we attempt to shed some light on a subset of the processes that appear to be required to realize these cognitive abilities. Starting from an abstract definition of what resembles a cognitive technical system (CTS), and which task it must achieve, we analyze an exemplary implementation and try to identify a modularized architecture that encompasses abstraction and separation between inference and decision making. We draw parallels between our interpretation of a general CTS and the architectures that have been used in already matured fields, such as automatic speech recognition, robotics and dialogue systems. We discuss implications, and possible challenges.

2 The Home Theater Setup Task

Throughout this chapter we will use the example application of an intelligent system whose function is to assist a human user with the setup of the devices of a home theater system. We have been involved in the implementation of such a system [4, 5], and its technical aspects are described in Chapter 24. We begin by describing the task the user tries to solve, and go on to describe the required properties of a technical system that provides assistance.

In the home cinema setup task (HCST), a person is faced with the problem of correctly connecting cables between various audio/video devices, such as TV, AV-amplifier, satellite receiver or DVD-player, using a multitude of available cables (Figure 1). The goal is to enable certain functionality, such as being able to watch DVDs on the TV, and to watch satellite TV. This goal can be achieved by correctly connecting the cables.



(a) Picture of two real devices.

(b) Instance with four devices and three cables.

Fig. 1: A picture of two typical devices, and a symbolic representation of an instance of the Home Theater Setup Task.

The HCST contains various difficulties for the involved human:

- It is necessary to have a mental model of the devices to predict what a specific cable configuration achieves.
- There exists a combinatorial problem when the number of available cables is limited, or when the number of connectors of the devices is low.
- Different solutions can potentially result in different trade-offs, like being able to turn off the DVD-player while watching satellite TV in one configuration, but not in another one.
- If the final setup is not working, the source of the error has to be diagnosed.

By providing aide to a human user during the HCST, some of these problems can be mitigated. We assume that an assistive CTS provides its functionality merely by communicating with the user. In order to solve the HCST efficiently and satisfactorily, the system requires at least the following abilities:

- knowledge about the problem domain, ports and functionality of the devices, connectors of cables, etc.
- a means to solve the combinatorial problem of identifying a connection scheme
- the ability to communicate with the user to provide directions and acquire feedback about the current state of the devices and cables

In addition the following can provide an enhanced experience to the user:

- knowledge about the user: Which commands can he be trusted with executing correctly? How to shape the communication, and how to choose a good granularity for the provided instructions.
- sensors to detect the current state of the environment automatically: this comprises the available devices, cables, and the desired goal

The implemented prototype was also able to handle a scenario with multiple persons. This included identifying the dedicated user who has to be communicated the instructions, and who is supposed to connect the devices. In the examined scenario, the system was able to localize persons using a laser range finder. Since output of

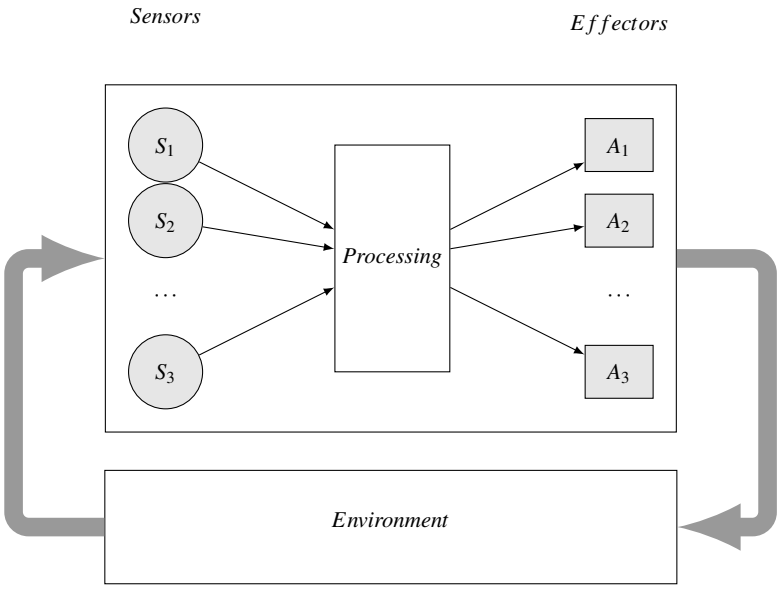


Fig. 2: A very abstract view of a CTS. The environment is perceived through a set of sensors and can be influenced via a set of effectors. This view leads to the interpretation as a POMDP with factored observation and action spaces.

the system could be routed through multiple display devices, the system must try to leverage output close to the dedicated user to minimize cost of communication. The same also holds for input devices in the case of touch screens.

3 Problem Statement

At a very abstract level, the function of a CTS can be reduced to the operation of a set of effectors, while observing data obtained through a set of sensors. The following formalization is closely related to Partially Observable Markov Decision Processs (POMDPs) with factored observations and actions. That relation is formalized in Section 3.2.

We are going to assume that the whole system is time-discrete with a shared, synchronized heart-beat. When ignoring the computational problem of acting intelligently, the possibilities of a CTS are limited by its hardware realization: a set of sensors $\mathscr{S}$, and a set of effectors $\mathscr{E}$. From this point of view, the system is supposed to interact with its environment by triggering the effectors depending on the information that it acquires through its sensors (Figure 2).

Each *sensor* $S \in \mathscr{S}$ is associated with a (not necessarily finite) set of possible observations Z^S. It produces a time-indexed sequence of observations z_t^S. The

observation-sequence is not determined in advance, but depends on the stochastic evolution of, and the interaction with the system's environment. Classical sensors of computer systems are keyboard and mouse. Modern smart phones have access to a much larger array of sensors, including but not limited to touch displays, accelerometers, cameras, and GPS. Access to the internet can also be treated as a sensor of some kind, as a computer is able to query various online information sources such as e-mail servers, postings of friends at social networks, or the weather forecast for next week.

An effector $A \in \mathscr{E}$ is associated with a (not necessarily finite) set of possible actions O^A. It has to be supplied with an action o_t^A for every time step t. Examples of possible effectors of an assistive CTS can be displays or speakers, as can be found with ordinary computer systems and smart phones. The role of these effectors is usually communication with the user. But the triggering of events external to the technical system can also be modelled as effectors. These can include posting an e-mail, issuing a job to a printer, or even initiating a purchase at some online market platform.

The task of a CTS at time t is to find actions o_t^A for every effector $A \in \mathscr{E}$, based on the past observations $z_{1:t-1}^S$ it has obtained through its sensors. The effectors shall be operated in an intelligent manner, maximizing the system's positive properties. In the context of *Companion* technology, these properties could be the *Companion* properties of individuality, adaptability, availability, co-operativeness and trustworthiness (cf. Chapter 1, [6]). Finding an operationalization for the "positive properties" of an assistive CTS is a difficult problem that is not to be discussed in this chapter. Independent of the concrete nature of these desirable properties, we can demand that the system acts in a way to optimize the expected value of a single given utility (or reward) function R that depends on the inaccessible state of the system's environment, and the action taken.

3.1 Markov Decision Processes

A Markov Decision Process (MDP) is a model for time-discrete, sequential decision problems within a fully observable, stochastic environment [3]. This means that actions can have an uncertain, probabilistic outcome. An agent that acts according to a MDP takes into account the possible failure of actions, and act in anticipation, possibly avoiding such actions on purpose. As such, solutions to MDPs deal with risk in a way that can be considered rational, by maximizing the attained expected utility [48].

For simplicity we restrict our attention to finite state and action spaces. Then an MDP is a tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, A is a finite set of actions, and $T\colon S \times A \to \mathscr{P}(S)$ maps state-action pairs to distributions over the following state[1]. In case the state space is continuous instead of finite, we talk of continu-

[1] $\mathscr{P}(X)$ is the set of all probability mass functions (or density functions) over the set X.

ous MDPs and the transition function is a probability density function. The reward $R\colon S \times A \to \mathbb{R}$ defines the immediate reward obtained from applying an action in some state.

A candidate solution for an MDP is called a *policy* $\pi\colon S \to A$ and maps states to actions. There exist different ways of defining the value of a policy. We present the traditional and most simple one—the discounted reward over an infinite horizon with discount factor γ. In this case, the expected discounted reward obtained under policy π when starting in state s is given by

$$J_\pi(s) = \mathbb{E}\left[R(s,\pi(s),s') + \gamma \cdot J_\pi(s')\right], \tag{1}$$

where, the expectation is over the successor state s' according to the transition distribution $T(s \mid s', \pi(s))$. Alternatives are the average reward over an infinite horizon [17], or the accumulated reward over a finite horizon [20]. The goal is to find a policy π that maximizes $J_\pi(s)$ for some initial state $s \in S$.

3.2 Partially Observable Markov Decision Processes

The model of POMDPs [20] further extends the MDP model by concealing the state of the environment from the acting agent. On each step, the agent is only provided with a limited observation that depends stochastically on the current state, but not with the state itself. Acting successfully within a POMDP can require to perform actions whose sole purpose is the acquisition of more information about the true state of the world.

Formally, a POMDP is a tuple $\langle S,A,T,O,Z,R\rangle$, where $\langle S,A,T,R\rangle$ is a MDP. Additionally, O is a finite set of observations, where the acting agent is given an observation $o \in O$ with probability $Z(o \mid s,a)$ after executing action a in (concealed) state s.

Since the agent does not have access to the true state, it has to base its decision on knowledge about past observations and applied actions. Thus a policy π for a POMDP is a mapping from a history of actions and observations $a_0, o_0, a_1, o_1, \cdots, a_{t-1}, a_{t-1}$ to the next action a_t. It can be shown, that policies for POMDPs can also be defined by mapping a probability distribution $b \in \mathscr{P}(S)$ over the state to actions [20]. Such a distribution is then called a *belief state*. A belief state can be updated to reflect the information gained after acting and obtaining a certain observation. A *belief update* on b when observing o after applying action a, is defined by the following formula:

$$b'(s) \propto Z(o \mid s,a) \sum_{s'} b(s') \cdot T(s \mid s',a) \tag{2}$$

Using this update, one can define a *belief MDP* that is equivalent to a given POMDP [20]. The belief MDP has the space of distributions over states $\mathscr{P}(S)$ as

its state space; it is thus a continuous state MDP. Thus, a policy for the belief MDP maps belief states to actions, and the value can be defined analogous to Equation 1.

A popular example of a POMDP is the tiger domain, where the agent faces two doors, behind one of which a dangerous tiger is waiting to devour the unwary (or unaware) agent. Solving the problem successfully involves listening closely to one of the doors; an action that does not influence the state, but yields an observation possibly revealing the true location of the beast. With full access to the world state, the listening action is pointless as the agent always knows which door is safe to enter.

While MDPs allow a planning agent to respond to stochastic/unexpected changes of the environment by using information gained from (perfectly) observing its state, solving a POMDP requires a *two-way* communication between the agent and the environment, which includes taking actions that elicit useful observations. It is thus not very surprising that dialogue systems [50, 49] and human-computer interaction [30] are one of the applications of POMDPs.

When looking at our initial formalization of CTSs by a set of senors $\mathscr{S}$ and a set of effectors $\mathscr{E}$, we can identify the sensors as the mean to obtaining observations within the POMDP framework. Joint assignments to the effectors form the space of possible actions. The POMDP framework fills the remaining gaps of our initial formalization—defining the system dynamics, the observation dynamics and the objective via a reward function.

The main difference lies within the fact that both observations and actions appear factored in our CTS formalization, e.g. $O = Z^{S_1} \times Z^{S_2} \times \cdots \times Z^{S_k}$ with $S_i \in \mathscr{S}$ and $A = O^{A_1} \times O^{A_2} \times \cdots \times O^{A_l}$ $A_i \in \mathscr{E}$. Factoring observation, state and action spaces is common for complex MDPs and POMDPs models [8, 41, 49].

4 A System Architecture—Divide and Conquer

When designing complex things like cars, robots or large software artifacts, we strive to conquer complexity by dividing the task into smaller, more manageable, parts. By defining what happens on the boundaries of those parts we create conceptual independence and enable working on one local problem at a time. By fixing the boundary conditions between modules we limit ourself to those solutions of the problem satisfying the boundary conditions. Usually the constraints will rule out the optimal solution, but they reduce the cognitive cost of designing the system and enable us to construct the system at all [9]. This form of modularization is one ingredient of this section.

Another one is constituted by the observation that *every way* of solving a problem—no matter how pragmatic it is—also solves an instance of the idealized problem. For us the idealized problem will be a complex POMDP, or a CTS problem, exemplarily capturing the HCST. We have "solved" this problem by building a working prototype [5] (see also Chapter 24). The prototype solves the combinatorial aspect of the HCST using a deterministic hybrid planning approach (Chapter 5).

Actions decided on by the planner are passed on to a dialogue manager that implements them by means of issuing instructions to the user. The dialogue manager passes on communication directives to a fission component that distributes them over available output channels.

While our system solves a task whose complete formalization requires the expressiveness of POMDPs, we have never formulated a joint problem, as this would be difficult. Somehow we shy away from modelling the environment, including a human user, as a single, hand-crafted Markov process, because of the many unknowns. But we have implicitly encoded a lot of assumptions about human communication behaviour within a dialogue manager [29] (and Chapter 9) and a multi-modal fusion and fission approach (Chapter 10). We formalized the technical aspects of the HCST as a planning problem (see Chapters 5 and 6). So we have solved many subproblems, but the solution to the idealized joint problem lies hidden between pragmatic decisions and implementations.

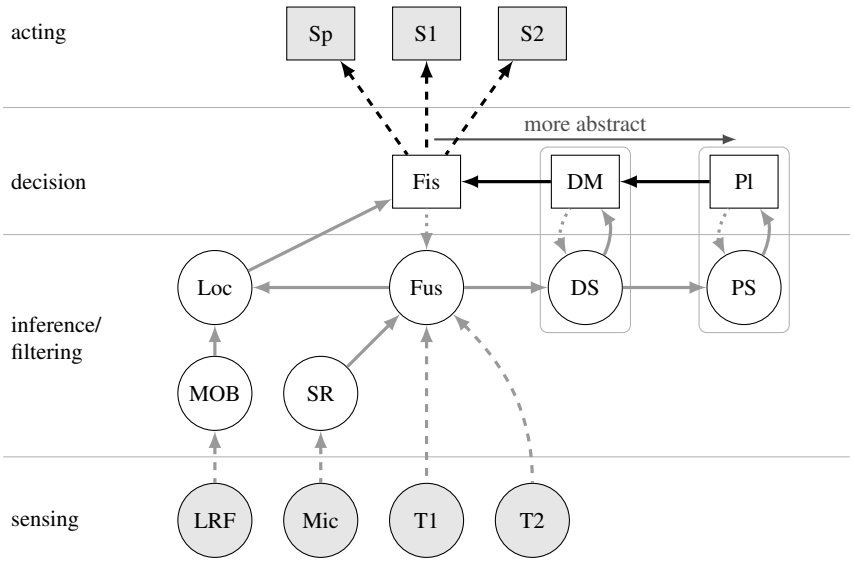


Fig. 3: Data-flow within a prototype implementation of a Cognitive Technical System

Figure 3 gives a model-based view on the internal workings of the prototype implementation (Chapter 24). On the lowest layer we have the sensors (from left to right: laser range finder (LRF), microphone (Mic), touch screens (T1, T2)). On the next layer components are shown that have the purpose of tracking the current state of the environment. The data from the LRF is processed by a multi-object bayes filter (MOB). Object localization data together with information from the input fusion (Fus) is used to locate the user (Loc). Audio is processed by a stock speech recognizer (SR), whose output is then processed by the multi-modal input fusion. The

current dialogue state (DS) uses the output of the input fusion, and provides information required to track the state of plan execution (PS). The belief state produced by the filtering stage is used to achieve decisions on how to control the effectors. The dedicated planner (Pl), the dialogue manager (DM), and the multi-modal output fission stage (Fis) are decision modules deciding upon actions. The decisions of the planner and the dialogue manager are only used to control the policy of other decision component, with only the fission being in direct control of effectors (speakers (Sp) and screens (S1, S2)). The type of data that flows between parts/components of the model is completely determined by the region boundaries crossed by the arrows. Observations flow from sensors to the filtering stage (◄- -). Nodes within the filtering stage produce marginal distributions over the complete belief state (◄—). The actions taken by the decision components get fed into the filtering stage as observations for the next time step (◄···). Outputs of decision components are either abstract, internal actions when used by another decision component (◄—), or primitive actions that control effector (◄- -).

The rest of this chapter is dedicated to the interpretation and generalization of the described prototype implementation.

5 Inference/Filtering

The observations that are produced by the sensors are consumed by model parts that incorporate this data into their probabilistic prediction of the current world state. This process of estimating the current world state based on past observation is sometimes called filtering, and algorithms implementing this process are called filters. This process is equivalent to the belief update for belief MDPs given in Equation 2. The nature of the components found within the filtering stage of Figure 3 requires some explanation. First, we can identify components that are true probabilistic filters. The laser-range-finder data is processed by a multi-object Bayes filter (see Chapter 15, [39]) to track objects near the system. Input from the speakers is processed by a stock automatic speech recognizer (SR). Speech recognition can be performed using hidden Markov models, which are themselves temporal probabilistic models with latent variables [12]. Fusion of input events (Fus) is performed using an approach based on the transferable belief model [43] (Chapter 10, [42]), and is thus able to treat uncertainty. Information about user interaction is fused with the tracking results obtained from (MOB) to identify a possible user among tracked objects (Loc) [15]. This is achieved using a high-level temporal, probabilistic model formulated in Markov logic [40, 14]. All the model parts of the filtering stage mentioned so far are either truly probabilistic filters, or at least use a different form of uncertainty management, as in the case of input fusion.

The planner (Pl) and the dialogue manager (DM) work with models that assume full observability, equivalent to solving MDP problems. The planner even follows a deterministic modelling approach without stochastic change of the environment, although it is able to handle unforeseen events by performing replanning [4]. Both

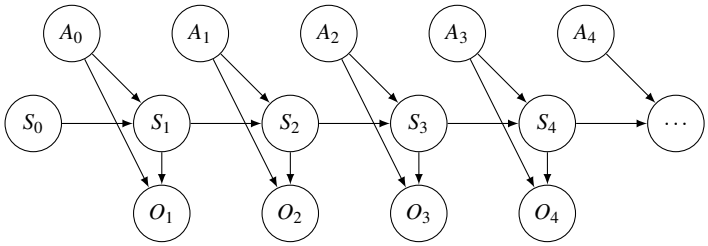


Fig. 4: Belief update for a POMDP interpreted as a dynamic Bayesian network [8], or a hidden Markov model. The prior distribution over the initial state S_0 is given by the initial belief state b_0.

modules have in common that they can be thought of tracking the current state using a deterministic finite automaton (DFA) (represented by DS, PS in Figure 3). In the prototype, changes to the planning model, which captures which cables have been connected to which devices, are tracked by observing the acknowledgement by the user of having executed a given instruction. Since a DFA can be considered a special case of a probabilistic temporal filter (one for a totally deterministic model and no uncertainty about the initial state), tracking of dialogue and planner state can be realized via Equation 2, too, and thus fits our interpretation.

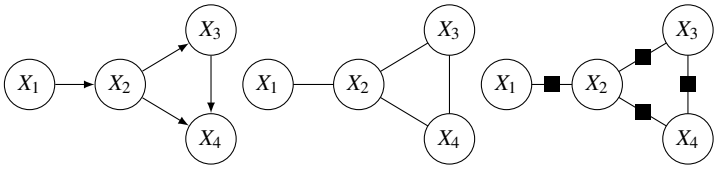
The product of a filter component consists of a probability distribution for the current state. Models of different components cover different variables, and the overlap between those variables is represented by arrows in Figure 3. This probabilistic knowledge is passed along the black arrows, which can be thought of as marginalizations of the distributions represented by their source. These marginal distributions are relevant either for other filters or for the decision modules within the next layer.

5.1 Factorizing the Belief State Using Graphical Models

As we have argued, the task that has to be performed jointly by the filtering modules in Figure 3 is to calculate the current belief. If we expand Equation 2 recursively to compute the belief b_t at time t, and define the initial belief state as b_0, we obtain

$$b_t(s_t \mid o_1, \ldots, o_t) = \sum_{s_0, \ldots, s_{t-1}} b_0(s_0) \prod_{i=1}^{t} T(s_i \mid s_{i-1}, a_{i-1}) Z(o_i \mid s_i, a_{i-1}). \qquad (3)$$

We can observe that the filtering stage needs to perform a marginalization over the past states $s_0, \ldots, s_{t-1}$ for a probability distribution that is given in factored form by the conditional probabilities T and Z (see Figure 4). This factorization corresponds to a hidden Markov model, or more generally a dynamic Bayesian network [28, 21].



(a) Bayesian network (b) Markov network (c) Factor graph

Fig. 5: Different flavors of graphical models. The Bayesian network shows the directed graph for distributions expressible through the factorization $P(X_1,\ldots,X_4) = P(X_1)\cdot P(X_2 \mid X_1)\cdot P(X_3 \mid X_2)\cdot P(X_4 \mid X_2,X_3)$. The Markov network and the factor graph are two different undirected graphical representations of the distribution defined by the factorization $P(X_1,\ldots,X_4) \propto \phi_1(X_1,X_2)\cdot\phi_2(X_2,X_3)\cdot\phi_3(x_3,x_4)\cdot\phi_4(x_4,x_2)$.

Multivariate probability distributions that are defined in a factored form are commonly known under the name of *probabilistic graphical models (PGMs)* [21]. The factorization of the distribution implies a set of stochastic (conditional) independencies that both reduce the required number of parameters to specify a distribution, and simplify probabilistic inference. PGMs come in two major flavors, namely Bayesian networks (BNs) and Markov random fields (MRFs). They differ in the type of used graphical representation (directed vs. undirected) and the type of factors they consist of. BNs consist of conditional probability tables while MRFs can consist of arbitrary non-negative multivariate functions.

BNs have been proposed for use in the area of Artificial Intelligence by Judea Pearl in the 1980s [34], although their roots range further back. A distribution P over variables $\mathbf{X} = \{X_1,\ldots,X_n\}$ represented by a BN is defined as the product of conditional probabilities (called conditional probability tables). A conditional probability table $P(X_i \mid \mathrm{Par}(X_i))$ encodes the distribution over variable X_i given the ancestor variables $\mathrm{Par}(X_i)$ in a directed acyclic graph that defines the dependency structure of the BN (see Figure 5a). The joint distribution is given by

$$P(X_1,\ldots,X_n) = \prod_i P(X_i \mid \mathrm{Par}(X_i)). \tag{4}$$

The undirected version of PGMs is called MRF or Markov network (MN) [21]. MRFs can be seen as a generalization of BNs where the conditional probabilities are replaced by arbitrary functions. The dependency structure of MRFs can be represented graphically either by an undirected graph (Markov network) or a factor graph ([22]; see Figure 5). A MRF over a set of variables $\mathbf{X} = \{X_1,\ldots,X_n\}$ is defined by a set of factors ϕ_a with index set $A \ni a$. A factor maps an assignment to a subset of variables $\mathbf{X}_a \subset \mathbf{X}$ to the positive reals, and the distribution is defined by the product over all factors:

$$P(X_1,\ldots,X_n) = \frac{1}{Z}\prod_{a\in A}\phi_a(\mathbf{X}_a) \tag{5}$$

As MRFs are not normalized, turning them into a proper distribution requires scaling by a normalizing constant Z called partition function with

$$Z = \sum_{X_1,\ldots,X_n}\prod_{a\in A}\phi_a(\mathbf{X}_a). \tag{6}$$

We have already observed that the joint filtering task consists of a marginalization in a BN whose conditional probability tables are given by the transition and observation probabilities (cf. Equation 3). These CPTs can be further broken down by assuming that the state S is factorized, just like observations and actions are factorized for the CTS task (cf. Section 3). This *inner* factorization enables the distributed filtering as can be seen in Figure 3. The factorization can be either of a directed or undirected nature, and we shall discuss the of the trade-offs between the two approaches now.

5.2 Markov Networks vs. Bayesian Networks

The conditional independence structures that can be represented by BNs and MNs are slightly different [34, pp. 126], although both models are able to represent any probability distribution over a finite set of discrete-valued random variables.

Causal models are more faithfully represented by BNs, as the independence structure expressible by directed edges allows for a causal interpretation [35]. Under the assumption that every dependence between random variables is due to a causal mechanism [37, p. 156], it is reasonable to assume that a POMDP (in particular T and Z) can be expressed in a factorized form using only directed edges. But since the marginal abstraction over variables in a BN can lead to undirected dependencies, i.e., spurious correlations that are caused by latent confounding variables, it is reasonable to assume that high-level models abstract over enough variables such that some dependencies cannot be considered causal anymore.

Since dependencies along the progression of time are usually of a causal nature, it is reasonable to represent transition probabilities T as directed dependency. Indeed it has been argued that undirected graphical models are not suited to describe temporal data [33, 18], although there are successful applications of temporal MRFs for labeling and segmentation of time series data [45].

An advantage of MRFs is their relatively unproblematic composability. Two MRFs over the same variables can easily be combined by multiplying them. The combination of BNs is not as painless, because one has to make sure that the resulting directed graph must be free of cycles. In addition MRFs can be regarded as a generalization of constraint satisfaction problems and boolean formulas in conjunctive normal form. This makes the use of existing deterministic formalizations trivial.

There exist numerous flavours of graphical models for the representation of time series data, such as maximum entropy Markov models [26], conditional random fields [23], or slice-normaliced dynamic Markov logic networks [33]. All models are addressing various shortcoming of the others. Chain graphs [24, 21] are a hybrid between BNs and MRFs, as they pose as a directed model on the high level, but allow conditional probability tables to be factorized in an undirected fashion. This allows to model the temporal progression between time steps as causal, while the interaction between variables of the same time step can be undirected. Thus, chain graphs pose as a potentially useful modelling paradigm in the context of CTSs.

6 Decision

The filtered estimates of the current state are used by the decision components to determine the behavior of the system, namely control of the effectors. In our interpretation of the prototype, there exist three distinguishable decision components.

At the lowest level we have the multi-modal fission (Fis). Fis is given abstract descriptions of dialogues and turns these into a concrete visual or audio output. For each information item that is to be conveyed to the user, Fis has to determine a modality (text, speech, image, video, etc.) and concrete output devices. For selection of a most suited device, Fis uses probabilistic information about the location of the target user (Loc) with the intent of choosing devices close to the user. Fis makes its decision according to the principle of maximizing expected utility [48]. The chosen output realization must be published to the multi-modal fusion (Fus), as e.g., pointing gestures can only be interpreted with knowledge about the on-screen location of involved entities. This is in accordance with our observation that the past action is necessary to update the current belief (Equation 2). Fis receives its control input (abstract dialogue descriptions) from the dialogue manager (DM).

The DM realizes communicative acts that may result in several dialogue turns each. An exemplary communicative act could be about issuing instructions to the user on connecting two devices with an available video-capable cable. The communicative act could start with querying the user about the availability of some suitable cables, and then instructing the user to plug in a user-chosen cable correctly. The implementation of the DM can be thought of as a POMDP policy that is predefined by an expert, instead of being obtained by planning/optimization. It follows that the DM component consumes user input to update its internal state, and it issues dialogue actions. For Figure 3 we have divided the dialogue manager into the policy (DM) and the state tracker (DS), though this separation was not manifest in the real implementation.

At the highest decision instance we employed a deterministic planner (Pl, see Chapter 5). The deterministic planning component uses replanning to handle the case when the observed state deviates from the expected state trajectory. This construction turns the planner into a MDP planner/policy, although it cannot anticipate risky situations. To close the gap between MDP and POMDP expressiveness, we

used the most probable state for detecting plan failures—a construction which only works in settings where the relevant transitions are nearly deterministic. This approach corresponds to the *belief replanning* paradigm in [11]. As for the DM, we have divided this replanning capable planner into a state tracker PS and the policy Pl. The state tracker has to maintain merely a most probable current world state, instead of a more general probabilistic belief state.

Following our architectural interpretation—where we have separated the tracking of the current state (filtering) into the inference stage—decision modules base their judgement solely on this belief state. This separation follows from the idea of the belief MDP for solving POMDP problems. And the equivalence between POMDP and belief MDP implies that separating filtering from decision making allows us to still obtain the optimal solution. To calculate the correct belief update, the inference stage requires knowledge of past actions and past sensory observations (cf. Equation 2). Notably, the separation results in purely functional (as in functional programming) decision components that do not rely on internal state that changes over time.

7 Abstraction

Within the prototype the only decision module that was interfacing with the effectors was the multi-modal fission (Chapter 10). It was given abstract decisions from the dialogue manager, which in turn was given abstract decisions from the planner. While it is perceivable that DM and/or Pl are also issuing commands to effectors directly, the configuration we find in the prototype appears to be common in other CTSs, as we shall see in Section 8. We are now going to discuss the types of abstractions we can identify within the prototype.

The abstraction in the prototype between Fis, DM and Pl is of two kinds. We can identify *temporal* abstraction in the sense that decisions made on higher levels have a longer duration and occur less frequently. In addition, a *arbitrational* abstraction reduces the size/dimensionality of the decision space towards the Pl end. In an extreme sense the lowest layer has to assign a color to every pixel of a screen, while on the higher level, between DM and Pl, the action is merely to “instruct the user to connect cable X with device Y”.

7.1 Temporal Abstraction

Temporal abstraction is a well researched topic in reinforcement learning [36, 19, 27, 7, 47], a field that deals with solving POMDP problems through interaction with the environment.[2] Within the MDP setting, options [46] formalize time-extended

[2] This is also called model-free reinforcement learning, as the model has to be learned together with a policy.

actions that are equipped with specialized sub-policies. The abstract decision propagation between decision components in the prototype bears much resemblance to options, as, e.g., the actions decided upon by the planner are implemented by the dialogue manager by a sequence of dialogues. A peculiarity of options is that the abstract policy cannot look inside options, and it is not able to interrupt them. This property also holds in both abstraction stages within the prototype. Because they have to choose between different actions less frequently, the higher level decision components are also able to use more complex algorithms that show a slower response. Decisions on the lower levels have to be found quickly, and thus it is not possible to take into account much context—they are often of a reactive character.

When examining Figure 3, we can observe a hierarchy of filtering models, mirroring the hierarchical arrangement of the decision components with the pairings Fus-Fis, DS-DM, and PS-Pl. Apparently it is natural to have abstraction within the state space of the joint POMDP, too. In contrast to hierarchical abstraction during decision making, the literature on temporal abstraction of probabilistic models appears to be much sparser. Brendan Burns and others describe an approach to temporal abstraction in Dynamic Bayesian networks (DBNs) [10]. Several works by Kalia Orphanou [31, 32] address temporal abstraction in DBNs for the evaluation of experimental data. Another very well researched field that strongly relies on temporal abstraction is automatic speech recognition, where approaches already span the range form the waveform over phonemes and words to the grammar level [12].

7.2 Arbitrational Abstraction

We have introduced the term *arbitrational* abstraction to describe the constellation where the action space or state is coarsened without changing the temporal extent. This concept on its own appears to be much less researched than temporal abstraction for action abstraction. This is potentially due to the fact that both types often occur together and thus are not recognized as separate phenomena, at least in the setting of planning and decision making. In [41] an approach to reinforcement learning with factored state and action spaces is described, where planning occurs by inference within a restricted Boltzmann machine. In the analysis, Sallans and Hinton identify some variables that act as “macro actions”. Their activation defines the joint behavior of many primitive decision variables. They also observe that the variables acting as macro actions are activated over longer periods of time, and thus arbitrational abstraction coincides with temporal abstraction again.

For probabilistic inference (not necessarily the temporal kind), arbitrational abstraction is omnipresent. The task of classification consists in mapping a high-dimensional input, for example an image, to a single output variable that represents an abstract concept like the proposition “contains a face”. Only recently, larger progress has been made in the field of *deep learning* [1], where classification is done over a sequence of layers. This layering of classificators has proven to yield a good boost in classification performance to earlier “shallow” architectures, and the used

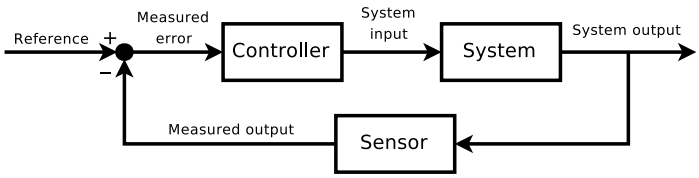


Fig. 6: The difference between a control input and a measurement of the target are used to control the dynamic system in a closed-loop architecture.

layer model can also be a probabilistic graphical model. As such, the crafted abstraction hierarchy on the inference side of Figure 3 bears some resemblance to "learned" abstraction hierarchies found by deep learning approaches.

8 Related Work

There are many examples of existing complex systems that can be thought of as CTSs. There exist problem areas where the complete issue of acting based on observations has to be (and to some extent *is*) solved; for example, robotics and dialogue management.

8.1 Control Theory

Control theory is concerned with the control of systems with simple dynamics. It has a long tradition and rests on solid mathematical foundations [2]. At the heart of the field lies the idea of exploiting feedback from measurements of the controlled system (see Figure 6). The simple closed-loop controller can be interpreted within our framework as having a degenerate filtering stage without latent variables. On the more complex end of control theory, the concept of multiple-input/multiple-output controllers [16, pp. 591] approaches the complexity of POMDPs with factored actions and observations.

8.2 Robot Architectures

It has been observed that many independently designed robot architectures have a similar structure. In robotics, three-layer architectures are common [13]. They consist of a *controller*, a *sequencer* and a *liberator*. The controller consists of computations that realize a tight coupling between sensors and effectors, and it is often using approaches from control theory [44, 16]. The sequencer determines the behavior

the controller is to follow, and possibly supplies additional parameters. According to [13], the sequencer resembles an MDP policy. On the highest level, the liberator implements computations that may require an arbitrary amount of processing time and cannot implement any real-time requirements. Our rendition of the general three-layer architecture is presented in Figure 7.

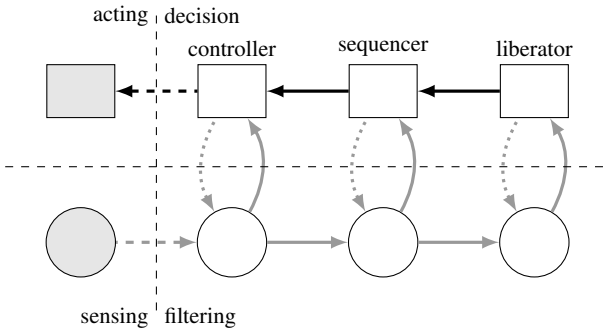


Fig. 7: Interpretation of the general three-layer-architecture.

8.3 Dialogue Systems

Multi-layer architectures have been used for dialogue management. For example [25] proposes a two layer architecture. The task of the lower layer is to react to phenomena related to maintenance of the communication channel, such as turn taking, back-channel feedback (from the listener to the speaker), or interruption. The higher layer is concerned with structuring and planning of the conversation. There are also attempts at tracking the dialogue state using graphical models [38].

9 Conclusion

We have analyzed a prototypical implementation of an assistive CTS. We have provided a descriptive problem specification by arguing that the joint behavior of the system can be considered an attempt to solve a POMDP problem. Based on the POMDP formalization, we have interpreted implemented software components as contributions of solving the POMDP using the belief MDP approach, thus partitioning functionality into filtering and decision-making. We have identified temporal and arbitrational abstraction as major components of the architecture. We have also discussed approaches to the modularization of filtering and decision making, and given references to related work.

Acknowledgements This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "*Companion*-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning-a new frontier in artificial intelligence research [research frontier]. Computational Intelligence Magazine, IEEE **5**(4), 13–18 (2010)
2. Åström, K.J., Kumar, P.: Control: A perspective. Automatica **50**(1), 3–43 (2014)
3. Bellman, R.: A markovian decision process. Tech. rep., DTIC Document (1957)
4. Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: Plan, repair, execute, explain - how planning helps to assemble your home theater. In: Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014), pp. 386–394. AAAI Press (2014)
5. Bercher, P., Richter, F., Hörnle, T., Geier, T., Höller, D., Behnke, G., Nothdurft, F., Honold, F., Minker, W., Weber, M., Biundo, S.: A planning-based assistance system for setting up a home theater. In: Proceedings of the 29th National Conference on Artificial Intelligence (AAAI 2015). AAAI Press (2015)
6. Biundo, S., Wendemuth, A.: *Companion*-technology for cognitive technical systems. Künstliche Intelligenz **30**(1), 71–75 (2016). DOI 10.1007/s13218-015-0414-8
7. Botvinick, M.M.: Hierarchical reinforcement learning and decision making. Current opinion in neurobiology **22**(6), 956–962 (2012)
8. Boutilier, C., Dean, T.L., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. J. Artif. Intell. Res. (JAIR) **11**, 1–94 (1999). DOI 10.1613/jair.575
9. Brusoni, S., Marengo, L., Prencipe, A., Valente, M.: The value and costs of modularity: A cognitive perspective. SPRU, SEWPS (2004)
10. Burns, B., Morrison, C.T.: Temporal abstraction in bayesian networks. In: AAAI Spring Symposium. Defense Technical Information Center (2003)
11. Cassandra, A.R., Kaelbling, L.P., Kurien, J.: Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS 1996, November 4-8, 1996, Osaka, Japan, pp. 963–972 (1996). DOI 10.1109/IROS.1996.571080
12. Gales, M., Young, S.: The application of hidden markov models in speech recognition. Foundations and Trends in Signal Processing **1**(3), 195–304 (2008)
13. Gat, E., et al.: On three-layer architectures. Artificial intelligence and mobile robots **195**, 210 (1998)
14. Geier, T., Biundo, S.: Approximate online inference for dynamic markov logic networks. In: International IEEE Conference on Tools with Artificial Intelligence, pp. 764 –768 (2011)
15. Geier, T., Reuter, S., Dietmayer, K., Biundo, S.: Goal-based person tracking using a first-order probabilistic model. In: Proceedings of the Nineth UAI Bayesian Modeling Applications Workshop (2012)
16. Goodwin, G.C., Graebe, S.F., Salgado, M.E.: Control System Design. Prentice Hall (2001)
17. Gosavi, A.: Reinforcement learning: A tutorial survey and recent advances. INFORMS Journal on Computing **21**(2), 178–192 (2009)
18. Jain, D., Barthels, A., Beetz, M.: Adaptive markov logic networks: Learning statistical relational models with dynamic parameters. In: ECAI, pp. 937–942 (2010)
19. Jong, N.K., Hester, T., Stone, P.: The utility of temporal abstraction in reinforcement learning. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '08, pp. 299–306. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2008)

20. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial intelligence **101**(1), 99–134 (1998)
21. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
22. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on **47**(2), 498 –519 (2001). DOI 10.1109/18.910572
23. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning (2001)
24. Lauritzen, S.L., Richardson, T.S.: Chain graph models and their causal interpretations. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **64**(3), 321–348 (2002)
25. Lemon, O., Cavedon, L., Kelly, B.: Managing dialogue interaction: A multi-layered approach. In: Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue, pp. 168–177 (2003)
26. McCallum, A., Freitag, D., Pereira, F.C.N.: Maximum entropy markov models for information extraction and segmentation. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000, pp. 591–598 (2000)
27. Montani, S., Bottrighi, A., Leonardi, G., Portinale, L.: A cbr-based, closed-loop architecture for temporal abstractions configuration. Computational Intelligence **25**(3), 235–249 (2009). DOI 10.1111/j.1467-8640.2009.00340.x
28. Murphy, K.: Dynamic bayesian networks: Representation, inference and learning. Ph.D. thesis, University of California (2002)
29. Nothdurft, F., Honold, F., Zablotskaya, K., Diab, A., Minker, W.: Application of verbal intelligence in dialog systems for multimodal interaction. In: Intelligent Environments (IE), 2014 International Conference on, pp. 361–364. IEEE (2014)
30. Nothdurft, F., Richter, F., Minker, W.: Probabilistic human-computer trust handling. In: 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, p. 51 (2014)
31. Orphanou, K., Keravnou, E., Moutiris, J.: Integration of Temporal Abstraction and Dynamic Bayesian Networks in Clinical Systems. A preliminary approach. In: A.V. Jones (ed.) 2012 Imperial College Computing Student Workshop, *OpenAccess Series in Informatics (OASIcs)*, vol. 28, pp. 102–108. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2012). DOI http://dx.doi.org/10.4230/OASIcs.ICCSW.2012.102
32. Orphanou, K., Stassopoulou, A., Keravnou, E.: Temporal abstraction and temporal bayesian networks in clinical domains: A survey. Artificial Intelligence in Medicine **60**(3), 133 – 149 (2014). DOI http://dx.doi.org/10.1016/j.artmed.2013.12.007
33. Papai, T., Kautz, H., Stefankovic, D.: Slice normalized dynamic markov logic networks. In: F. Pereira, C. Burges, L. Bottou, K. Weinberger (eds.) Advances in Neural Information Processing Systems 25, pp. 1907–1915. Curran Associates, Inc. (2012)
34. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
35. Pearl, J.: Causality: models, reasoning and inference, vol. 29. Cambridge Univ Press (2000)
36. Rafols, E., Koop, A., Sutton, R.S.: Temporal abstraction in temporal-difference networks. In: Y. Weiss, B. Schölkopf, J. Platt (eds.) Advances in Neural Information Processing Systems 18, pp. 1313–1320. MIT Press (2006)
37. Reichenbach, H., Reichenbach, M.: The Direction of Time. Philosophy (University of California (Los Ángeles))). University of California Press (1991)
38. Ren, H., Xu, W., Zhang, Y., Yan, Y.: Dialog state tracking using conditional random fields. In: Proceedings of the SIGDIAL 2013 Conference, pp. 457–461. Association for Computational Linguistics, Metz, France (2013)
39. Reuter, S., Dietmayer, K.: Pedestrian tracking using random finite sets. In: Proceedings of the 14th International Conference on Information Fusion, pp. 1–8 (2011)
40. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2), 107–136 (2006)
41. Sallans, B., Hinton, G.E.: Reinforcement learning with factored states and actions. The Journal of Machine Learning Research **5**, 1063–1088 (2004)

42. Schüssel, F., Honold, F., Weber, M.: Using the transferable belief model for multimodal input fusion in companion systems. In: Multimodal Pattern Recognition of Social Signals in Human-Computer-Interaction, pp. 100–115. Springer (2013)
43. Smets, P., Kennes, R.: The transferable belief model. Artificial intelligence **66**(2), 191–234 (1994)
44. Sontag, E.D.: Mathematical control theory: deterministic finite dimensional systems, vol. 6. Springer Science & Business Media (1998)
45. Sutton, C., McCallum, A., Rohanimanesh, K.: Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. The Journal of Machine Learning Research **8**, 693–723 (2007)
46. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence **112**(1), 181–211 (1999)
47. Theocharous, G., Kaelbling, L.P.: Approximate planning in pomdps with macro-actions. In: S. Thrun, L. Saul, B. Schölkopf (eds.) Advances in Neural Information Processing Systems 16, pp. 775–782. MIT Press (2004)
48. Von Neumann, J., Morgenstern, O.: Theory of games and economic behavior. Princeton University Press (1944)
49. Williams, J.D., Poupart, P., Young, S.: Factored partially observable markov decision processes for dialogue management. In: 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI), pp. 76–82 (2005)
50. Young, S., Gasic, M., Thomson, B., Williams, J.D.: Pomdp-based statistical spoken dialog systems: A review. Proceedings of the IEEE **101**(5), 1160–1179 (2013)