Journal of Artificial General Intelligence 1 (2009) 3-24 Submitted 2009-01-09; Revised 2009-06-09
DOI: 10.2478/v10229-011-0002-8

Feature Reinforcement Learning: Part I. Unstructured MDPs[*]

Marcus Hutter MARCUS@HUTTER1.NET
RSISE @ ANU and SML @ NICTA WWW.HUTTER1.NET
CANBERRA, ACT, 0200, AUSTRALIA

Editor: Itamar Arel

Abstract

General-purpose, intelligent, learning agents cycle through sequences of observations, actions, and rewards that are complex, uncertain, unknown, and non-Markovian. On the other hand, reinforcement learning is well-developed for small finite state Markov decision processes (MDPs). Up to now, extracting the right state representations out of bare observations, that is, reducing the general agent setup to the MDP framework, is an art that involves significant effort by designers. The primary goal of this work is to automate the reduction process and thereby significantly expand the scope of many existing reinforcement learning algorithms and the agents that employ them. Before we can think of mechanizing this search for suitable MDPs, we need a formal objective criterion. The main contribution of this article is to develop such a criterion. I also integrate the various parts into one learning algorithm. Extensions to more realistic dynamic Bayesian networks are developed in Part II (Hutter, 2009c). The role of POMDPs is also considered there.

Keywords: Reinforcement learning; Markov decision process; partial observability; feature learning; explore-exploit; information & complexity; rational agents.

"Approximations, after all, may be made in two places - in the construction of the model and in the solution of the associated equations. It is not at all clear which yields a more judicious approximation."

— *Richard Bellman (1961)*

1. Introduction

Background & motivation. Artificial General Intelligence (AGI) is concerned with designing agents that perform well in a wide range of environments (Goertzel and Pennachin, 2007; Legg and Hutter, 2007). Among the well-established "narrow" Artificial Intelligence (AI) approaches (Russell and Norvig, 2003), arguably Reinforcement Learning (RL) (Sutton and Barto, 1998) pursues most directly the same goal. RL considers the general agent-environment setup in which an agent interacts with an environment (acts and observes in cycles) and receives (occasional) rewards. The agent's objective is to collect as much reward as possible. Most if not all AI problems can be formulated in this framework. Since the future is generally unknown and uncertain, the agent needs to learn a model of the

[*]. A shorter version appeared in the proceedings of the AGI 2009 conference (Hutter, 2009b).

environment based on past experience, which allows to predict future rewards and use this to maximize expected long-term reward.

The simplest interesting environmental class consists of finite state fully observable Markov Decision Processes (MDPs) (Puterman, 1994; Sutton and Barto, 1998), which is reasonably well understood. Extensions to continuous states with (non)linear function approximation (Sutton and Barto, 1998; Gordon, 1999), partial observability (POMDP) (Kaelbling, Littman, and Cassandra, 1998; Ross et al., 2008), structured MDPs (DBNs) (Strehl, Diuk, and Littman, 2007), and others have been considered, but the algorithms are much more brittle.

A way to tackle complex real-world problems is to reduce them to finite MDPs which we know how to deal with efficiently. This approach leaves a lot of work to the designer, namely to extract the right state representation ("features") out of the bare observations in the initial (formal or informal) problem description. Even if *potentially* useful representations have been found, it is usually not clear which ones will turn out to be better, except in situations where we already know a perfect model. Think of a mobile robot equipped with a camera plunged into an unknown environment. While we can imagine which image features will potentially be useful, we cannot know in advance which ones will actually be useful.

Main contribution. The primary goal of this paper is to develop and investigate a method that *automatically* selects those features that are necessary and sufficient for *reducing* a complex real-world problem to a computationally tractable MDP.

Formally, we consider maps Φ from the past observation-reward-action history h of the agent to an MDP state. Histories not worth being distinguished are mapped to the same state, i.e. Φ^{-1} induces a partition on the set of histories. We call this model ΦMDP. A state may be simply an abstract label of the partition, but more often is itself a structured object like a discrete vector. Each vector component describes one feature of the history (Hutter, 2009a,c). For example, the state may be a 3-vector containing (shape,color,size) of the object a robot tracks. For this reason, we call the *reduction*, *Feature RL*, although in this Part I only the simpler unstructured case is considered.

Φ maps the agent's experience over time into a sequence of MDP states. Rather than informally constructing Φ by hand, our goal is to develop a formal objective criterion $\text{Cost}(\Phi|h)$ for *evaluating* different reductions Φ. Obviously, at any point in time, if we want the criterion to be effective it can only depend on the agent's past experience h and possibly generic background knowledge. The "Cost" of Φ shall be small iff it leads to a "good" MDP representation. The establishment of such a criterion transforms the, in general, ill-defined RL problem to a formal optimization problem (minimizing Cost) for which efficient algorithms need to be developed. Another important question is which problems *can* profitably be reduced to MDPs (Hutter, 2009a,c).

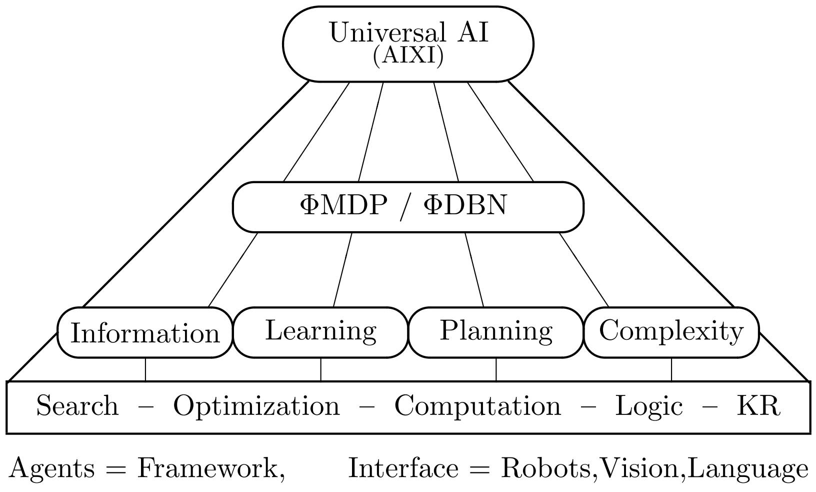
The real world does not conform itself to nice models: Reality is a non-ergodic partially observable uncertain unknown environment in which acquiring experience can be expensive. So we should exploit the data (past experience) at hand as well as possible, cannot generate virtual samples since the model is not given (need to be learned itself), and there is no reset-option. No criterion for this general setup exists. Of course, there is previous work which is in one or another way related to ΦMDP.

ΦMDP in perspective. As partly detailed later, the suggested ΦMDP model has interesting connections to many important ideas and approaches in RL and beyond:

- ΦMDP side-steps the open problem of learning POMDPs (Kaelbling, Littman, and Cassandra, 1998),
- Unlike Bayesian RL algorithms (Dearden, Friedman, and Andre, 1999; Duff, 2002; Poupart et al., 2006; Ross and Pineau, 2008), ΦMDP avoids learning a (complete stochastic) observation model,
- ΦMDP is a scaled-down practical instantiation of AIXI (Hutter, 2005, 2007),
- ΦMDP extends the idea of state-aggregation from planning (based on bi-simulation metrics (Givan, Dean, and Greig, 2003)) to RL (based on information),
- ΦMDP generalizes U-Tree (McCallum, 1996) to arbitrary features,
- ΦMDP extends model selection criteria to general RL problems (Grünwald, 2007),
- ΦMDP is an alternative to PSRs (Singh et al., 2003) for which proper learning algorithms have yet to be developed,
- ΦMDP extends feature selection from supervised learning to RL (Guyon and Elisseeff, 2003).

Learning in agents via rewards is a much more demanding task than "classical" machine learning on independently and identically distributed (i.i.d.) data, largely due to the temporal credit assignment and exploration problem. Nevertheless, RL (and the closely related adaptive control theory in engineering) has been applied (often unrivaled) to a variety of real-world problems, occasionally with stunning success (Backgammon, Checkers, (Sutton and Barto, 1998, Chp.11), helicopter control (Ng et al., 2004)). ΦMDP overcomes several of the limitations of the approaches in the items above and thus broadens the applicability of RL.

ΦMDP owes its general-purpose *learning* and *planning* ability to its *information* and *complexity* theoretical foundations. The implementation of ΦMDP is based on (specialized and general) *search* and *optimization* algorithms used for finding good reductions Φ. Given that ΦMDP aims at general AI problems, one may wonder about the role of other aspects traditionally considered in AI (Russell and Norvig, 2003): *knowledge representation* (KR) and *logic* may be useful for representing complex reductions $\Phi(h)$. Agent interface fields like *robotics*, computer *vision*, and natural *language* processing can speedup learning by pre&post-processing the raw observations and actions into more structured formats. These representational and interface aspects will only barely be discussed in this paper. The following diagram illustrates ΦMDP in perspective.



Contents. Section 2 formalizes our ΦMDP setup, which consists of the agent model with a map Φ from observation-reward-action histories to MDP states. Section 3 develops our core Φ selection principle, which is illustrated in Section 4 on a tiny example. Section 5 discusses general search algorithms for finding (approximations of) the optimal Φ, concretized for context tree MDPs. In Section 6 I find the optimal action for ΦMDP, and present the overall algorithm. Section 7 improves the Φ selection criterion by "integrating" out the states. Section 8 contains a brief discussion of ΦMDP, including relations to prior work, incremental algorithms, and an outlook to more realistic *structured* MDPs (dynamic Bayesian networks, ΦDBN) treated in Part II.

Rather than leaving parts of ΦMDP vague and unspecified, I decided to give at the very least a simplistic concrete algorithm for each building block, which may be assembled to one sound system on which one can build on.

Notation. Throughout this article, log denotes the binary logarithm, ϵ the empty string, and $\delta_{x,y}=\delta_{xy}=1$ if $x=y$ and 0 else is the Kronecker symbol. I generally omit separating commas if no confusion arises, in particular in indices. For any x of suitable type (string,vector,set), I define string $\boldsymbol{x}=x_{1:l}=x_1...x_l$, sum $x_+=\sum_j x_j$, union $x_*=\bigcup_j x_j$, and vector $\boldsymbol{x}_\bullet=(x_1,...,x_l)$, where j ranges over the full range $\{1,...,l\}$ and $l=|\mathrm{x}|$ is the length or dimension or size of x. $\hat{x}$ denotes an estimate of x. $\mathrm{P}(\cdot)$ denotes a probability over states and rewards or parts thereof. I do not distinguish between random variables X and realizations x, and abbreviation $\mathrm{P}(x):=\mathrm{P}[X=x]$ never leads to confusion. More specifically, $m\in\mathbb{N}$ denotes the number of states, $i\in\{1,...,m\}$ any state index, $n\in\mathbb{N}$ the current time, and $t\in\{1,...,n\}$ any time in history. Further, in order not to get distracted at several places I gloss over initial conditions or special cases where inessential. Also 0∗undefined=0∗infinity:=0.

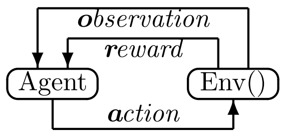
2. Feature Markov Decision Process (ΦMDP)

This section describes our formal setup. It consists of the agent-environment framework and maps Φ from observation-reward-action histories to MDP states. I call this arrangement "Feature MDP" or short ΦMDP.

Agent-environment setup. I consider the standard agent-environment setup (Russell and Norvig, 2003) in which an *Agent* interacts with an *Environment* The agent can choose from actions $a\in\mathcal{A}$ (e.g. limb movements) and the environment provides (regular) observations $o\in\mathcal{O}$ (e.g. camera images) and real-valued rewards $r\in\mathcal{R}\subseteq I\!R$ to the agent. The reward may be very scarce, e.g. just $+1$ (-1) for winning (losing) a chess game, and 0 at all other times (Hutter, 2005, Sec.6.3). This happens in cycles $t=1,2,3,...$: At time t, after observing o_t and receiving reward r_t, the agent takes action a_t based on history $h_t:=o_1r_1a_1...o_{t-1}r_{t-1}a_{t-1}o_tr_t$. Then the next cycle $t+1$ starts. The agent's objective is to maximize his long-term reward. Without much loss of generality, I assume that $\mathcal{R}$ is finite. Finiteness of $\mathcal{R}$ is lifted in Hutter (2009a,c). I also assume that $\mathcal{A}$ is finite and small, which is restrictive. Part II deals with large state spaces, and large (structured) action spaces can be dealt with in a similar way. No assumptions are made on $\mathcal{O}$; it may be huge or even infinite. Indeed, ΦMDP has been specifically designed to cope with huge observation spaces, e.g. camera images, which are mapped to a small space of relevant states.

The agent and environment may be viewed as a pair or triple of interlocking functions of the history $\mathcal{H}:=(\mathcal{O}\times\mathcal{R}\times\mathcal{A})^*\times\mathcal{O}\times\mathcal{R}$:

$$\begin{aligned}\text{Env}:\mathcal{H}\times\mathcal{A}\rightsquigarrow\mathcal{O}\times\mathcal{R}, &\quad o_nr_n=\text{Env}(h_{n-1}a_{n-1}),\\ \text{Agent}:\mathcal{H}\rightsquigarrow\mathcal{A}, &\quad a_n=\text{Agent}(h_n),\end{aligned}$$



where $\rightsquigarrow$ indicates that mappings $\to$ might be stochastic.

The goal of AI is to design agents that achieve high (expected) reward over the agent's lifetime.

(Un)known environments. For known Env(), finding the reward maximizing agent is a well-defined and formally solvable problem (Hutter, 2005, Chp.4), with computational efficiency being the “only” matter of concern. For most real-world AI problems Env() is at best partially known. For unknown Env(), the meaning of expected reward maximizing is even conceptually a challenge (Hutter, 2005, Chp.5).

Narrow AI considers the case where function Env() is either known (like planning in blocks world), or essentially known (like in chess, where one can safely model the opponent as a perfect minimax player), or Env() belongs to a relatively small class of environments (e.g. elevator or traffic control).

The goal of AGI is to design agents that perform well in a large range of environments (Legg and Hutter, 2007), i.e. achieve high reward over their lifetime with as little as possible assumptions about Env(). A minimal necessary assumption is that the environment possesses *some* structure or pattern (Wolpert and Macready, 1997).

From real-life experience (and from the examples below) we know that usually we do not need to know the complete history of events in order to determine (sufficiently well) what will happen next and to be able to perform well. Let $\Phi(h)$ be such a “useful” summary of history h.

Generality of ΦMDP. The following examples show that many problems can be reduced (approximately) to finite MDPs, thus showing that ΦMDP can deal with a large variety of problems: In full-information *games* (like chess) with a static opponent, it is sufficient to know the current state of the game (board configuration) to play well (the history plays no role), hence $\Phi(h_t)=o_t$ is a sufficient summary (Markov condition). Classical *physics* is essentially predictable from the position and velocity of objects at a single time, or equivalently from the locations at two consecutive times, hence $\Phi(h_t) = o_{t-1}o_t$ is a sufficient summary (2nd order Markov). For *i.i.d. processes* of unknown probability (e.g. clinical trials $\simeq$ Bandits), the frequency of observations $\Phi(h_n) = (\sum_{t=1}^n \delta_{o_t o})_{o\in\mathcal{O}}$ is a sufficient statistic. In a *POMDP planning* problem, the so-called belief vector at time t can be written down explicitly as some function of the complete history h_t (by integrating out the hidden states). $\Phi(h_t)$ could be chosen as (a discretized version of) this belief vector, showing that ΦMDP generalizes POMDPs. Obviously, the *identity* $\Phi(h) = h$ is always sufficient but not very useful, since Env() as a function of $\mathcal{H}$ is hard to impossible to "learn".

This suggests to look for Φ with small codomain, which allow to learn/estimate/approximate Env by $\widehat{\text{Env}}$ such that $o_t r_t \approx \widehat{\text{Env}}(\Phi(h_{t-1}))$ for $t = 1...n$.

Example. Consider a robot equipped with a camera, i.e. o is a pixel image. Computer vision algorithms usually extract a set of features from o_{t-1} (or h_{t-1}), from low-level patterns to high-level objects with their spatial relation. Neither is it possible nor necessary to make a precise prediction of o_t from summary $\Phi(h_{t-1})$. An approximate prediction must and will do. The difficulty is that the similarity measure "$\approx$" needs to be context dependent. Minor image nuances are irrelevant when driving a car, but when buying a painting it makes a huge difference in price whether it's an original or a copy. Essentially only a bijection Φ would be able to extract *all potentially* interesting features, but such a Φ defeats its original purpose.

From histories to states. It is of utmost importance to properly formalize the meaning of "$\approx$" in a general, domain-independent way. Let $s_t := \Phi(h_t)$ summarize all relevant information in history h_t. I call s a state or feature (vector) of h. "Relevant" means that the future is predictable from s_t (and a_t) alone, and that the relevant future is coded in $s_{t+1}s_{t+2}...$. So we pass from the complete (and known) history $o_1 r_1 a_1 ... o_n r_n a_n$ to a "compressed" history $sra_{1:n} \equiv s_1 r_1 a_1 ... s_n r_n a_n$ and seek Φ such that s_{t+1} is (approximately a stochastic) function of s_t (and a_t). Since the goal of the agent is to maximize his rewards, the rewards r_t are always relevant, so they (have to) stay untouched (this will become clearer below).

The ΦMDP. The structure derived above is a classical Markov Decision Process (MDP), but the primary question I ask is not the usual one of finding the value function or best action or comparing different models of a given state sequence. I ask how well can the state-action-reward sequence generated by Φ be modeled as an MDP compared to other sequences resulting from different Φ. A good Φ leads to a good model for predicting future rewards, which can be used to find good actions that maximize the agent's expected long-term reward.

3. ΦMDP Coding and Evaluation

I first review a few standard codes and model selection methods for i.i.d. sequences, subsequently adapt them to our situation, and show that they are suitable in our context. I state my Cost function for Φ, and the Φ selection principle, and compare it to the Minimum Description Length (MDL) philosophy.

I.i.d. processes. Consider i.i.d. $x_1...x_n \in \mathcal{X}^n$ for finite $\mathcal{X}=\{1,...,m\}$. For known $\theta_i = \mathrm{P}[x_t=i]$ we have $\mathrm{P}(x_{1:n}|\boldsymbol{\theta}) = \theta_{x_1}\cdot...\cdot\theta_{x_n}$. It is well-known that there exists a code (e.g. arithmetic or Shannon-Fano) for $x_{1:n}$ of length $-\log\mathrm{P}(x_{1:n}|\boldsymbol{\theta})$, which is asymptotically optimal with probability one (Barron, 1985, Thm.3.1). This also easily follows from (Cover and Thomas, 2006, Thm.5.10.1).

MDL/MML code (Grünwald, 2007; Wallace, 2005): For unknown $\boldsymbol{\theta}$ we may use a frequency estimate $\hat{\theta}_i = n_i/n$, where $n_i = |\{t \leq n : x_t = i\}|$. Then it is easy to see that $-\log\mathrm{P}(x_{1:n}|\hat{\boldsymbol{\theta}}) = n\,H(\hat{\boldsymbol{\theta}})$, where

$$H(\hat{\boldsymbol{\theta}}) \;:=\; -\sum_{i=1}^{m}\hat{\theta}_i\log\hat{\theta}_i \quad \text{is the entropy of} \quad \hat{\boldsymbol{\theta}}$$

($0\log 0 := 0 =: 0\log\frac{0}{0}$). We also need to code $\hat{\boldsymbol{\theta}}$, or equivalently (n_i), which naively needs $\log n$ bits for each i. In general, a sample size of n allows estimating parameters only to accuracy $O(1/\sqrt{n})$, which is essentially equivalent to the fact that $\log\mathrm{P}(x_{1:n}|\hat{\boldsymbol{\theta}} \pm O(1/\sqrt{n})) - \log\mathrm{P}(x_{1:n}|\hat{\boldsymbol{\theta}}) = O(1)$. This shows that it is sufficient to code each $\hat{\theta}_i$ to accuracy $O(1/\sqrt{n})$, which requires only $\frac{1}{2}\log n + O(1)$ bits each. Hence, given n and ignoring $O(1)$ terms, the overall code length (CL) of $x_{1:n}$ for unknown frequencies is

$$\mathrm{CL}(x_{1:n}) \;\equiv\; \mathrm{CL}(\boldsymbol{n}) \;:=\; n\,H(\boldsymbol{n}/n) + \tfrac{m-1}{2}\log n \quad \text{for} \quad n>0 \quad \text{and} \quad 0 \quad \text{else}, \tag{1}$$

where $\boldsymbol{n} = (n_1,...,n_m)$ and $n = n_+ = n_1 + ... + n_m$. We have assumed that n is given, hence only $m-1$ of the n_i need to be coded, since the mth one can be reconstructed from them and n. The above is an exact code of $x_{1:n}$, which is optimal (within $+O(1)$) for all i.i.d. sources. This code may further be optimized by only coding $\hat{\theta}_i$ for the $m' = |\{i : n_i > 0\}| \leq m$ non-empty categories, resulting in a code of length

$$\mathrm{CL}'(\boldsymbol{n}) \;:=\; n\,H(\boldsymbol{n}/n) + \tfrac{m'-1}{2}\log n + m, \tag{2}$$

where the m bits are needed to indicate which of the $\hat{\theta}_i$ are coded. We refer to this improvement as *sparse* code.

Combinatorial code (Li and Vitányi, 2008): A second way to code the data is to code $\boldsymbol{n}$ exactly, and then, since there are $n!/n_1!...n_m!$ sequences $x_{1:n}$ with counts $\boldsymbol{n}$, we can easily construct a code of length $\log(n!/n_1!...n_m!)$ given $\boldsymbol{n}$ by enumeration, i.e.

$$\mathrm{CL}''(\boldsymbol{n}) \;:=\; \log(n!/n_1!...n_m!) + (m-1)\log n$$

Within $\pm O(1)$ this code length also coincides with (1).

Incremental code (Willems, Shtarkov, and Tjalkens, 1997): A third way is to use a sequential estimate $\hat{\theta}_i^{t+1} = \frac{t_i+\alpha}{t+m\alpha}$ based on *known* past counts $t_i = |\{t' \leq t : x_{t'} = i\}|$, where $\alpha > 0$ is some regularizer. Then

$$\mathrm{P}(x_{1:n}) \;=\; \hat{\theta}_{x_1}^1 \cdot ... \cdot \hat{\theta}_{x_n}^n \;=\; C_\alpha \frac{\prod_{i=1}^m \Gamma(n_i+\alpha)}{\Gamma(n+m\alpha)}, \qquad C_\alpha := \frac{\Gamma(m\alpha)}{\Gamma(\alpha)^m} \tag{3}$$

where Γ is the Gamma function. The logarithm of this expression again essentially reduces to (1) (for any $\alpha > 0$, typically $\frac{1}{2}$ or 1), which can also be written as

$$\mathrm{CL}'''(\boldsymbol{n}) = \sum_{i:n_i>0} \ln\Gamma(n_i) - \ln\Gamma(n) + O(1) \quad \text{if} \quad n>0 \quad \text{and } 0 \text{ else.}$$

Bayesian code (Schwarz, 1978; MacKay, 2003): A fourth (the Bayesian) way is to assume a Dirichlet(α) prior over $\boldsymbol{\theta}$. The marginal distribution (evidence) is identical to (3) and the Bayesian Information Criterion (BIC) approximation leads to code (1).

Conclusion: All four methods lead to essentially the same code length. The references above contain rigorous derivations. In the following I will ignore the $O(1)$ terms and refer to (1) simply as *the* code length. Note that $x_{1:n}$ is coded exactly (lossless). Similarly (see MDP below) sampling models more complex than i.i.d. may be considered, and the one that leads to the shortest code is selected as the best model (Grünwald, 2007).

MDP definitions. Recall that a sequence $sra_{1:n}$ is said to be sampled from an MDP $(\mathcal{S},\mathcal{A},T,R)$ iff the probability of s_t only depends on s_{t-1} and a_{t-1}; and r_t only on s_{t-1}, a_{t-1}, and s_t. That is,

$$\begin{aligned} \mathrm{P}(s_t|h_{t-1}a_{t-1}) &= \mathrm{P}(s_t|s_{t-1},a_{t-1}) &=: T^{a_{t-1}}_{s_{t-1}s_t} \\ \mathrm{P}(r_t|h_t) &= \mathrm{P}(r_t|s_{t-1},a_{t-1},s_t) &=: R^{a_{t-1}r_t}_{s_{t-1}s_t} \end{aligned}$$

In our case, we can identify the state-space $\mathcal{S}$ with the states $s_1,...,s_n$ "observed" so far. Hence $\mathcal{S}=\{s^1,...,s^m\}$ is finite and typically $m\ll n$, since states repeat. Let $s\overset{a}{\to}s'(r')$ be shorthand for "action a in state s resulted in state s' (reward r')". Let $\mathcal{T}^{ar'}_{ss'}:=\{t\leq n: s_{t-1}=s,a_{t-1}=a,s_t=s',r_t=r'\}$ be the set of times $t-1$ at which $s\overset{a}{\to}s'r'$, and $n^{ar'}_{ss'}:=|\mathcal{T}^{ar'}_{ss'}|$ their number ($n^{++}_{++}=n$).

Coding MDP sequences. For some fixed s and a, consider the subsequence $s_{t_1}...s_{t_{n'}}$ of states reached from s via a ($s\overset{a}{\to}s_{t_i}$), i.e. $\{t_1,...,t_{n'}\}=\mathcal{T}^{a*}_{s*}$, where $n'=n^{a+}_{s+}$. By definition of an MDP, this sequence is i.i.d. with s' occurring $n'_{s'}:=n^{a+}_{ss'}$ times. By (1) we can code this sequence in $\mathrm{CL}(\boldsymbol{n}')$ bits. The whole sequence $s_{1:n}$ consists of $|\mathcal{S}\times\mathcal{A}|$ i.i.d. sequences, one for each $(s,a)\in\mathcal{S}\times\mathcal{A}$. We can join their codes and get a total code length

$$\mathrm{CL}(s_{1:n}|a_{1:n}) \;=\; \sum_{s,a}\mathrm{CL}(\boldsymbol{n}^{a+}_{s\bullet}) \tag{4}$$

If instead of (1) we use the improved sparse code (2), non-occurring transitions $s\overset{a}{\to}s'r'$ will contribute only one bit rather than $\frac{1}{2}\log n$ bits to the code, so that large but sparse MDPs get penalized less.

Similarly to the states we code the rewards. There are different "standard" reward models. I consider only the simplest case of a small discrete reward set $\mathcal{R}$ like $\{0,1\}$ or $\{-1,0,+1\}$ here and defer generalizations to $I\!R$ and a discussion of variants to the ΦDBN model (Hutter, 2009a). By the MDP assumption, for each (s,a,s') triple, the rewards at times $\mathcal{T}^{a*}_{ss'}$ are i.i.d. Hence they can be coded in

$$\mathrm{CL}(r_{1:n}|s_{1:n},a_{1:n}) \;=\; \sum_{s,a,s'}\mathrm{CL}(\boldsymbol{n}^{a\bullet}_{ss'}) \tag{5}$$

bits. In order to increase the statistics it might be better to treat r_t as a function of s_t only. This is not restrictive, since dependence on s_{t-1} and a_{t-1} can be mimicked by coding aspects into an enlarged state space.

Reward$\leftrightarrow$state trade-off. Note that the code for $\boldsymbol{r}$ depends on $\boldsymbol{s}$. Indeed we may interpret the construction as follows: Ultimately we/the agent cares about the reward, so we want to measure how well we can predict the rewards, which we do with (5). But this code depends on $\boldsymbol{s}$, so we need a code for $\boldsymbol{s}$ too, which is (4). To see that we need both parts consider two extremes.

A simplistic state transition model (small $|\mathcal{S}|$) results in a short code for $\boldsymbol{s}$. For instance, for $|\mathcal{S}|=1$, nothing needs to be coded and (4) is identically zero. But this obscures potential structure in the reward sequence, leading to a long code for $\boldsymbol{r}$.

On the other hand, the more detailed the state transition model (large $|\mathcal{S}|$) the easier it is to predict and hence compress $\boldsymbol{r}$. But a large model is hard to learn, i.e. the code for $\boldsymbol{s}$ will be large. For instance for $\Phi(h)=h$, no state repeats and the frequency-based coding breaks down.

Φ selection principle. Let us define the *Cost* of $\Phi:\mathcal{H}\to\mathcal{S}$ on h_n as the length of the ΦMDP code for $\boldsymbol{sr}$ given $\boldsymbol{a}$ plus a complexity penalty CL(Φ) for Φ:

$$\begin{aligned}\text{Cost}(\Phi|h_n) \;&:=\; \text{CL}(s_{1:n}|a_{1:n}) + \text{CL}(r_{1:n}|s_{1:n},a_{1:n}) + \text{CL}(\Phi), \\ \text{where} \quad s_t = \Phi(h_t) \quad &\text{and} \quad h_t = ora_{1:t-1}o_tr_t\end{aligned} \tag{6}$$

The discussion above suggests that the minimum of the joint code length (4) and (5) is attained for a Φ that keeps all and only relevant information for predicting rewards. Such a Φ may be regarded as best explaining the rewards. I added an additional complexity penalty CL(Φ) for Φ such that from the set of Φ that minimize (4)+(5) (e.g. Φ's identical on $(\mathcal{O}\times\mathcal{R}\times\mathcal{A})^n$ but different on longer histories) the simplest one is selected. The penalty is usually some code-length or log-index of Φ. This conforms with Ockham's razor and the MDL philosophy. So we are looking for a Φ of minimal cost:

$$\Phi^{best} \;:=\; \arg\min_{\Phi}\{\text{Cost}(\Phi|h_n)\} \tag{7}$$

If the minimization is restricted to some small class of reasonably simple Φ, CL(Φ) in (6) may be dropped. The state sequence generated by Φ^{best} (or approximations thereof) will usually only be approximately MDP. While Cost($\Phi|h$) is an optimal code only for MDP sequences, it still yields good codes for approximate MDP sequences. Indeed, Φ^{best} balances closeness to MDP with simplicity. The primary purpose of the simplicity bias is *not* computational tractability, but generalization ability (Legg, 2008; Hutter, 2005).

Relation to MDL et al. In unsupervised learning (clustering and density estimation) and supervised learning (regression and classification), penalized maximum likelihood criteria (Hastie, Tibshirani, and Friedman, 2001, Chp.7) like BIC (Schwarz, 1978), MDL (Grünwald, 2007), and MML (Wallace, 2005) have successfully been used for semi-parametric model selection. It is far from obvious how to apply them in RL. Indeed, our derived Cost function cannot be interpreted as a usual model+data code length. The problem is the following:

Ultimately we do not care about the observations but the rewards. The rewards depend on the states, but the states are arbitrary in the sense that they are model-dependent functions of the bare data (observations). The existence of these unobserved states is what complicates matters, but their introduction is necessary in order to model the rewards. For instance, Φ is actually not needed for coding $\boldsymbol{rs}|\boldsymbol{a}$, so from a strict coding/MDL perspective, CL(Φ) in (6) is redundant. Since $\boldsymbol{s}$ is some "arbitrary" construct of Φ, it is better to regard

(6) as a code of $\boldsymbol{r}$ only. Since the agent chooses his actions, $\boldsymbol{a}$ need not be coded, and $\boldsymbol{o}$ is not coded, because they are only of indirect importance.

The Cost() criterion is strongly motivated by the rigorous MDL principle, but invoked outside the usual induction/modeling/prediction context.

4. A Tiny Example

The purpose of the tiny example in this section is to provide enough insight into how and why ΦMDP works to convince the reader that our Φ selection principle is reasonable.

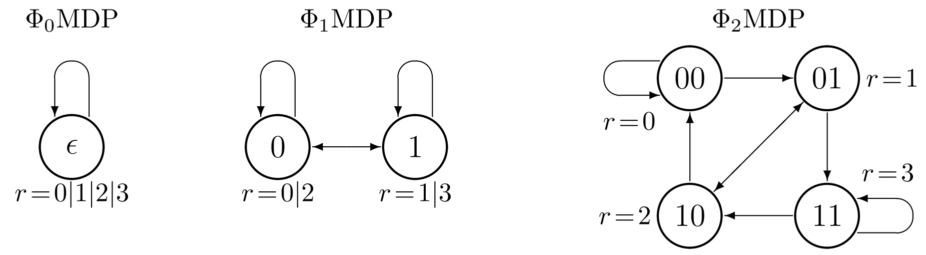
Example setup. I assume a simplified MDP model in which reward r_t only depends on s_t, i.e.

$$\text{CL}(r_{1:n}|s_{1:n},a_{1:n}) \;=\; \sum_{s'}\text{CL}(\boldsymbol{n}_{+s'}^{+\bullet}) \tag{8}$$

This allows us to illustrate ΦMDP on a tiny example. The same insight is gained using (5) if an analogous larger example is considered. Furthermore I set $\text{CL}(\Phi)\equiv 0$.

Consider binary observation space $\mathcal{O}=\{0,1\}$, quaternary reward space $\mathcal{R}=\{0,1,2,3\}$, and a single action $\mathcal{A}=\{0\}$. Observations o_t are independent fair coin flips, i.e. Bernoulli($\frac{1}{2}$), and reward $r_t=2o_{t-1}+o_t$ a deterministic function of the two most recent observations.

Considered features. As features Φ I consider $\Phi_k:\mathcal{H}\to\mathcal{O}^k$ with $\Phi_k(h_t)=o_{t-k+1}...o_t$ for various $k=0,1,2,...$ which regard the last k observations as "relevant". Intuitively Φ_2 is the best observation summary, which I confirm below. The state space $\mathcal{S}=\{0,1\}^k$ (for sufficiently large n). The ΦMDPs for $k=0,1,2$ are as follows.



Φ_2MDP with all non-zero transition probabilities being 50% is an exact representation of our data source. The missing arrow (directions) are due to the fact that $s=o_{t-1}o_t$ can only lead to $s'=o'_t o'_{t+1}$ for which $o'_t=o_t$, denoted by $s*=*s'$ in the following. Note that ΦMDP does not "know" this and has to learn the (non)zero transition probabilities. Each state has two successor states with equal probability, hence generates (see previous paragraph) a Bernoulli($\frac{1}{2}$) state subsequence and a constant reward sequence, since the reward can be computed from the state = last two observations. Asymptotically, all four states occur equally often, hence the sequences have approximately the same length $n/4$.

In general, if $\boldsymbol{s}$ (and similarly $\boldsymbol{r}$) consists of $x\in I\!N$ i.i.d. subsequences of equal length n/x over $y\in I\!N$ symbols, the code length (4) (and similarly (8)) is

$$\begin{aligned}\text{CL}(\boldsymbol{s}|\boldsymbol{a};x_y) &= n\log y + x\tfrac{|\mathcal{S}|-1}{2}\log\tfrac{n}{x}\\ \text{CL}(\boldsymbol{r}|\boldsymbol{s},\boldsymbol{a};x_y) &= n\log y + x\tfrac{|\mathcal{R}|-1}{2}\log\tfrac{n}{x}\end{aligned}$$

where the extra argument x_y just indicates the sequence property. So for Φ_2MDP we get

$$\text{CL}(\boldsymbol{s}|\boldsymbol{a};4_2) = n + 6\log\tfrac{n}{4} \quad\text{and}\quad \text{CL}(\boldsymbol{r}|\boldsymbol{s},\boldsymbol{a};4_1) = 6\log\tfrac{n}{4}$$

The log-terms reflect the required memory to code the MDP structure and probabilities. Since each state has only 2 realized/possible successors, we need n bits to code the state sequence. The reward is a deterministic function of the state, hence needs no memory to code given $\boldsymbol{s}$.

The Φ_0MDP throws away all observations (left figure above), hence $\text{CL}(\boldsymbol{s}|\boldsymbol{a};1_1)=0$. While the reward sequence is *not* i.i.d. (e.g. $r_{t+1}=3$ cannot follow $r_t=0$), Φ_0MDP has no choice regarding them as i.i.d., resulting in $\text{CL}(\boldsymbol{r}|\boldsymbol{a};1_4)=2n+\frac{3}{2}\log n$.

The Φ_1MDP model is an interesting compromise (middle figure above). The state allows a partial prediction of the reward: State 0 allows rewards 0 and 2; state 1 allows rewards 1 and 3. Each of the two states creates a Bernoulli($\frac{1}{2}$) state successor subsequence and a binary reward sequence, wrongly presumed to be Bernoulli($\frac{1}{2}$). Hence $\text{CL}(\boldsymbol{s}|\boldsymbol{a};2_2)=n+\log\frac{n}{2}$ and $\text{CL}(\boldsymbol{r}|\boldsymbol{s},\boldsymbol{a};2_2)=n+3\log\frac{n}{2}$.

Summary. The following table summarizes the results for general $k=0,1,2$ and beyond:

k	$\mathcal{S}$	$\lvert\mathcal{S}\rvert$	$n^{0+}_{ss'}$	$n^{+r'}_{+s'}$	$n^{0+}_{s+}=n^{++}_{+s'}$	$\boldsymbol{s}+\boldsymbol{r}$	$\text{CL}(\boldsymbol{s}\|\boldsymbol{a})$	$\text{CL}(\boldsymbol{r}\|\boldsymbol{s},\boldsymbol{a})$	$\text{Cost}(\Phi\|h)$
0	$\{\epsilon\}$	1	n	$n/4$	n	1_1+1_4	0	$2n+\frac{3}{2}\log n$	$2n+\frac{3}{2}\log n$
1	$\{0,1\}$	2	$n/4$	$\frac{n}{4}\delta_{r'-s'=0\|1}$	$n/2$	2_2+2_2	$n+\log\frac{n}{2}$	$n+3\log\frac{n}{2}$	$2n+4\log\frac{n}{2}$
2	$\left\{\begin{smallmatrix}00,01\\10,11\end{smallmatrix}\right\}$	4	$\frac{n}{8}\delta_{s*,*s'}$	$\frac{n}{4}\delta_{r'\hat{=}s'}$	$n/4$	4_2+4_1	$n+6\log\frac{n}{4}$	$6\log\frac{n}{4}$	$n+12\log\frac{n}{4}$
≥2	$\{0,1\}^k$	2^k	$\frac{n\delta_{s*,*s'}}{2^{k+1}}$	$\frac{n}{4}\delta_{r'\hat{=}s'}$	$n/2^k$	$2^k_2+2^k_1$	$n+\frac{2^k-1}{2^{1-k}}\log\frac{n}{2^k}$	$\frac{3}{2}2^k\log\frac{n}{2^k}$	$n+\frac{2^k+2}{2^{1-k}}\log\frac{n}{2^k}$

The notation of the $\boldsymbol{s}+\boldsymbol{r}$ column follows the one used above in the text (x_y for $\boldsymbol{s}$ and $\boldsymbol{r}$). $r'\hat{=}s'$ means that r' is the correct reward for state s'. The last column is the sum of the two preceding columns. The part linear in n is the code length for the state/reward sequence. The part logarithmic in n is the code length for the transition/reward probabilities of the MDP; each parameter needs $\frac{1}{2}\log n$ bits. For large n, Φ_2 results in the shortest code, as anticipated. The "approximate" model Φ_1 is just not good enough to beat the vacuous model Φ_0, but in more realistic examples some approximate model usually has the shortest code. Hutter (2009a) shows on a more complex example how Φ^{best} will store long-term information in a POMDP environment.

5. Cost(Φ) Minimization

So far I have reduced the reinforcement learning problem to a formal Φ-optimization problem. This section briefly explains what we have gained by this reduction, and provide some general information about problem representations, stochastic search, and Φ neighborhoods. Finally I present a simplistic but concrete algorithm for searching context tree MDPs.

Φ search. I now discuss how to find good summaries Φ. The introduced generic cost function $\text{Cost}(\Phi|h_n)$, based on only the known history h_n, makes this a well-defined task that is completely decoupled from the complex (ill-defined) reinforcement learning objective. This reduction should not be under-estimated. We can employ a wide range of optimizers and do not even have to worry about overfitting. The most challenging task is to come up with creative algorithms proposing Φ's.

There are many optimization methods: Most of them are search-based: random, blind, informed, adaptive, local, global, population based, exhaustive, heuristic, and other search methods (Aarts and Lenstra, 1997). Most are or can be adapted to the structure of the objective function, here $\text{Cost}(\cdot|h_n)$. Some exploit the structure more directly (e.g. gradient methods for convex functions). Only in very simple cases can the minimum be found analytically (without search).

Most search algorithms require the specification of a neighborhood relation or distance between candidate Φ, which I define in the 2nd next paragraph.

Problem representation can be important: Since Φ is a discrete function, searching through (a large subset of) all computable functions, is a non-restrictive approach. Variants of Levin search (Schmidhuber, 2004; Hutter, 2005) and genetic programming (Koza, 1992; Banzhaff et al., 1998) and recurrent neural networks (Pearlmutter, 1989; Raedt et al., 2008) are the major approaches in this direction.

A different representation is as follows: Φ effectively partitions the history space $\mathcal{H}$ and identifies each partition with a state. Conversely any partition of $\mathcal{H}$ can (up to a renaming of states) uniquely be characterized by a function Φ. Formally, Φ induces a (finite) partition $\bigcup_s\{h':\Phi(h')=s\}$ of $\mathcal{H}$, where s ranges over the codomain of Φ. Conversely, any partition of $\mathcal{H}=\mathcal{B}_1\dot\cup...\dot\cup\mathcal{B}_m$ induces a function $\Psi(h')=i$ iff $h'\in\mathcal{B}_i$, which is equivalent to Φ apart from an irrelevant permutation of the codomain (renaming of states).

State aggregation methods have been suggested earlier for solving large-scale MDP planning problems by grouping (partitioning) similar states together, resulting in (much) smaller block MDPs (Givan, Dean, and Greig, 2003). But the used bi-simulation metrics require knowledge of the MDP transition probabilities, while our Cost criterion does not.

Decision trees/lists/grids/etc. are essentially space partitioners. The most powerful versions are rule-based, in which logical expressions recursively divide domain $\mathcal{H}$ into "true/false" regions (Dzeroski, de Raedt, and Driessens, 2001; Sanner and Boutilier, 2009).

Φ neighborhood relation. A natural "minimal" change of a partition is to subdivide=split a partition or merge (two) partitions. Moving elements from one partition to another can be implemented as a split and merge operation. In our case this corresponds to splitting and merging states (state refinement and coarsening). Let Φ' split some state $s^a\in\mathcal{S}$ of Φ into $s^b,s^c\notin\mathcal{S}$

$$\Phi'(h) \;:=\; \begin{cases} \Phi(h) & \text{if} \quad \Phi(h)\neq s^a \\ s^b \text{ or } s^c & \text{if} \quad \Phi(h)=s^a \end{cases}$$

where the histories mapped to state s^a are distributed among s^b and s^c according to some splitting rule (e.g. randomly). The new state space is $\mathcal{S}'=\mathcal{S}\setminus\{s^a\}\cup\{s^b,s^c\}$. Similarly Φ' merges states $s^b,s^c\in\mathcal{S}$ into $s^a\notin\mathcal{S}$ if

$$\Phi'(h) \;:=\; \begin{cases} \Phi(h) & \text{if} \quad \Phi(h)\neq s^a \\ s^a & \text{if} \quad \Phi(h)=s^b \text{ or } s^c \end{cases}$$

where $\mathcal{S}'=\mathcal{S}\setminus\{s^b,s^c\}\cup\{s^s\}$. We can regard Φ' as being a neighbor of or similar to Φ.

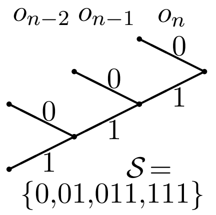
Stochastic Φ search. Stochastic search is the method of choice for high-dimensional unstructured problems. Monte Carlo methods can actually be highly effective, despite their simplicity (Liu, 2002; Fishman, 2003). The general idea is to randomly choose a neighbor Φ' of Φ and replace Φ by Φ' if it is better, i.e. has smaller Cost. Even if $\text{Cost}(\Phi'|h) > \text{Cost}(\Phi|h)$ we may keep Φ', but only with some (in the cost difference exponentially) small probability. Simulated annealing is a version which minimizes $\text{Cost}(\Phi|h)$. Apparently, Φ of small cost are (much) more likely to occur than high cost Φ.

Context tree example. The Φ_k in Section 4 depended on the last k observations. Let us generalize this to a context dependent variable length: Consider a finite complete suffix free set of strings (= prefix tree of reversed strings) $\mathcal{S} \subset \mathcal{O}^*$ as our state space (e.g. $\mathcal{S} = \{0,01,011,111\}$ for binary $\mathcal{O}$), and define $\Phi_\mathcal{S}(h_n) := s$ iff $o_{n-|s|+1:n} = s \in \mathcal{S}$, i.e. s is the part of the history regarded as relevant. State splitting and merging works as follows: For binary $\mathcal{O}$, if history part $s \in \mathcal{S}$ of h_n is deemed too short, we replace s by $0s$ and $1s$ in $\mathcal{S}$, i.e. $\mathcal{S}' = \mathcal{S} \setminus \{s\} \cup \{0s,1s\}$. If histories $1s,0s \in \mathcal{S}$ are deemed too long, we replace them by s, i.e. $\mathcal{S}' = \mathcal{S} \setminus \{0s,1s\} \cup \{s\}$. Large $\mathcal{O}$ might be coded binary and then treated similarly. For small $\mathcal{O}$ we have the following simple Φ-optimizer:

ΦImprove($\Phi_\mathcal{S}$,h_n)

```
⌈ Randomly choose a state $s \in \mathcal{S}$;
  Let $p$ and $q$ be uniform random numbers in [0,1];
  if ($p > 1/2$) then split $s$ i.e. $S' = S \setminus \{s\} \cup \{os : o \in \mathcal{O}\}$
  else if $\{os' : o \in \mathcal{O}\} \subseteq \mathcal{S}$     ($s'$ *is $s$ without the first symbol*)
  then merge them, i.e. $S' = S \setminus \{os' : o \in \mathcal{O}\} \cup \{s'\}$;
  if ($\text{Cost}(\Phi_\mathcal{S}|h_n) - \text{Cost}(\Phi_{\mathcal{S}'}|h_n) > \log(q)$) then $\mathcal{S} := \mathcal{S}'$;
⌊ return ($\Phi_\mathcal{S}$);
```

Example tree



The idea of using suffix trees as state space is from McCallum (1996) (see also Ring, 1994). It might be interesting to compare the local split/merge criterion of McCallum (1996) with our general global Cost criterion. On the other hand, due to their limitation, suffix trees are currently out of vogue.

6. Exploration & Exploitation

Having obtained a good estimate $\hat{\Phi}$ of Φ^{best} in the previous section, we can/must now determine a good action for our agent. For a finite MDP with known transition probabilities, finding the optimal action is routine. For estimated probabilities we run into the infamous exploration-exploitation problem, for which promising approximate solutions have recently been suggested by Szita and Lörincz (2008). At the end of this section I present the overall algorithm for our ΦMDP agent.

Optimal actions for known MDPs. For a known finite MDP $(\mathcal{S},\mathcal{A},T,R,\gamma)$, the maximal achievable ("optimal") expected future discounted reward sum, called (Q) Value (of action a) in state s, satisfies the following (Bellman) equations (Sutton and Barto, 1998)

$$Q_s^{*a} \;=\; \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma V_{s'}^*] \quad \text{and} \quad V_s^* = \max_a Q_s^{*a} \tag{9}$$

where $0<\gamma<1$ is a discount parameter, typically close to 1. See Hutter (2005, Sec.5.7) for proper choices. The equations can be solved by a simple (e.g. value or policy) iteration process or various other methods or in guaranteed polynomial time by dynamic programming (Puterman, 1994). The optimal next action is

$$a_n := \arg\max_a Q^{*a}_{s_n} \tag{10}$$

Estimating the MDP. We can estimate the transition probability T by

$$\hat{T}^a_{ss'} \;:=\; \frac{n^{a+}_{ss'}}{n^{a+}_{s+}} \quad \text{if} \quad n^{a+}_{s+}>0 \quad \text{and} \quad 0 \quad \text{else.} \tag{11}$$

It is easy to see that the Shannon-Fano code of $s_{1:n}$ based on $\mathrm{P}_{\hat{T}}(s_{1:n}|a_{1:n})=\prod_{t=1}^n \hat{T}^{a_{t-1}}_{s_{t-1}s_t}$ plus the code of the (non-zero) transition probabilities $\hat{T}^a_{ss'}$ to relevant accuracy $O(1/\sqrt{n^{a+}_{s+}})$ has length (4), i.e. the frequency estimate (11) is consistent with the attributed code length. The expected reward can be estimated as

$$\hat{R}^a_{ss'} := \sum_{r'\in\mathcal{R}} \hat{R}^{ar'}_{ss'} r', \qquad \hat{R}^{ar'}_{ss'} := \frac{n^{ar'}_{ss'}}{n^{a+}_{ss'}} \tag{12}$$

Exploration. Simply replacing T and R in (9) and (10) by their estimates (11) and (12) can lead to very poor behavior, since parts of the state space may never be explored, causing the estimates to stay poor.

Estimate $\hat{T}$ improves with increasing n^{a+}_{s+}, which can (only) be ensured by trying all actions a in all states s sufficiently often. But the greedy policy above has no incentive to explore, which may cause the agent to perform very poorly: The agent stays with what he *believes* to be optimal without trying to solidify his belief. For instance, if treatment A cured the first patient, and treatment B killed the second, the greedy agent will stick to treatment A and not explore the possibility that B may just have failed due to bad luck. Trading off exploration versus exploitation optimally is computationally intractable (Hutter, 2005; Poupart et al., 2006; Ross and Pineau, 2008) in all but extremely simple cases (e.g. Bandits, see Berry and Fristedt, 1985; Kumar and Varaiya, 1986). Recently, polynomially optimal algorithms (Rmax,E3,OIM) have been invented by Kearns and Singh (1998); Brafman and Tennenholtz (2002); Szita and Lörincz (2008) and others: An agent is more explorative if he expects a high reward in the unexplored regions. We can "deceive" the agent to believe this by adding another "absorbing" high-reward state s^e to $\mathcal{S}$, not in the range of $\Phi(h)$, i.e. never observed. Henceforth, $\mathcal{S}$ denotes the extended state space. For instance + in (11) now includes s^e. We set

$$n^a_{ss^e}=1, \quad n^a_{s^e s}=\delta_{s^e s}, \quad R^a_{ss^e}=R^e_{max} \tag{13}$$

for all s,a, where exploration bonus R^e_{max} is polynomially (in $(1-\gamma)^{-1}$ and $|\mathcal{S}\times\mathcal{A}|$) larger than $\max\mathcal{R}$ (Szita and Lörincz, 2008).

Now compute the agent's action by (9)-(12) but for the extended $\mathcal{S}$. The optimal policy p^* tries to find a chain of actions and states that likely leads to the high reward absorbing state s^e. Transition $\hat{T}^a_{ss^e}=1/n^a_{s+}$ is only "large" for small n^a_{s+}, hence p^* has a bias towards unexplored (state,action) regions. It can be shown that this algorithm makes only a polynomial number of sub-optimal actions.

The overall algorithm for our ΦMDP agent is as follows.

ΦMDP-Agent($\mathcal{A}$,$\mathcal{R}$)

```
⌈ Initialize $\Phi\equiv\Phi'\equiv\epsilon$; $\mathcal{S}=\{\epsilon\}$; $h_0=a_0=r_0=\epsilon$;
  for $n=1,2,3,...$
  ⌈ Choose e.g. $\gamma=1-1/n$;
    Set $R^e_{max}$=Polynomial($(1-\gamma)^{-1}$,$|\mathcal{S}\times\mathcal{A}|$)$\cdot$max$\mathcal{R}$;
    While waiting for $o_n$ and $r_n$
    ⌈ $\Phi':=\Phi$Improve($\Phi'$,$h_{n-1}$);
    ⌊ If Cost($\Phi'|h_{n-1}$)$<$Cost($\Phi|h_{n-1}$) then $\Phi:=\Phi'$;
    Observe $o_n$ and $r_n$; $h_n:=h_{n-1}a_{n-1}o_nr_n$;
    $s_n:=\Phi(h_n)$; $\mathcal{S}:=\mathcal{S}\cup\{s_n\}$;
    Compute action $a_n$ from Equations (9)-(13);
⌊ ⌊ Output action $a_n$;
```

7. Improved Cost Function

As discussed, we ultimately only care about (modeling) the rewards, but this endeavor required introducing and coding states. The resulting Cost($\Phi|h$) function is a code length of not only the rewards but also the "spurious" states. This likely leads to a too strong penalty of models Φ with large state spaces $\mathcal{S}$. The proper Bayesian formulation developed in this section allows to "integrate" out the states. This leads to a code for the rewards only, which better trades off accuracy of the reward model and state space size.

For an MDP with transition and reward probabilities $T^a_{ss'}$ and $R^{ar'}_{ss'}$, the probabilities of the state and reward sequences are

$$\mathrm{P}_T(s_{1:n}|a_{1:n}) = \prod_{t=1}^n T^{a_{t-1}}_{s_{t-1}s_t}, \quad \mathrm{P}_R(r_{1:n}|s_{1:n}a_{1:n}) = \prod_{t=1}^n R^{a_{t-1}r_t}_{s_{t-1}s_t}$$

The probability of $\boldsymbol{r}|\boldsymbol{a}$ can be obtained by taking the product and marginalizing $\boldsymbol{s}$:

$$\begin{aligned}\mathrm{P}_U(r_{1:n}|a_{1:n}) &= \sum_{s_{1:n}} \mathrm{P}_T(s_{1:n}|a_{1:n})\mathrm{P}_R(r_{1:n}|s_{1:n}a_{1:n}) \\ &= \sum_{s_{1:n}}\prod_{t=1}^n U^{a_{t-1}r_t}_{s_{t-1}s_t} = \sum_{s_n}[U^{a_0r_1}\cdots U^{a_{n-1}r_n}]_{s_0s_n}\end{aligned}$$

where for each $a\in\mathcal{A}$ and $r'\in\mathcal{R}$, matrix $U^{ar'}\in I\!R^{m\times m}$ is defined as $[U^{ar'}]_{ss'}\equiv U^{ar'}_{ss'}:=T^a_{ss'}R^{ar'}_{ss'}$. The right n-fold matrix product can be evaluated in time $O(m^2n)$. This shows that $\boldsymbol{r}$ given $\boldsymbol{a}$ and U can be coded in $-\log\mathrm{P}_U$ bits. The unknown U needs to be estimated, e.g. by the relative frequency $\hat{U}^{ar'}_{ss'}:=n^{ar'}_{ss'}/n^{a+}_{s+}$. Note that P_U completely ignores the observations $o_{1:n}$ and is essentially independent of Φ. Map Φ and hence $o_{1:n}$ enter $\mathrm{P}_{\hat{U}}$ (only and crucially) via the estimate $\hat{U}$. The $M:=m(m-1)|\mathcal{A}|(|\mathcal{R}|-1)$ (independent) elements of $\hat{U}$ can be coded to sufficient accuracy in $\frac{1}{2}M\log n$ bits, and Φ will be coded in CL(Φ) bits. Together this leads to a code for $\boldsymbol{r}|\boldsymbol{a}$ of length

$$\mathrm{ICost}(\Phi|h_n) := -\log\mathrm{P}_{\hat{U}}(r_{1:n}|a_{1:n}) + \tfrac{1}{2}M\log n + \mathrm{CL}(\Phi) \tag{14}$$

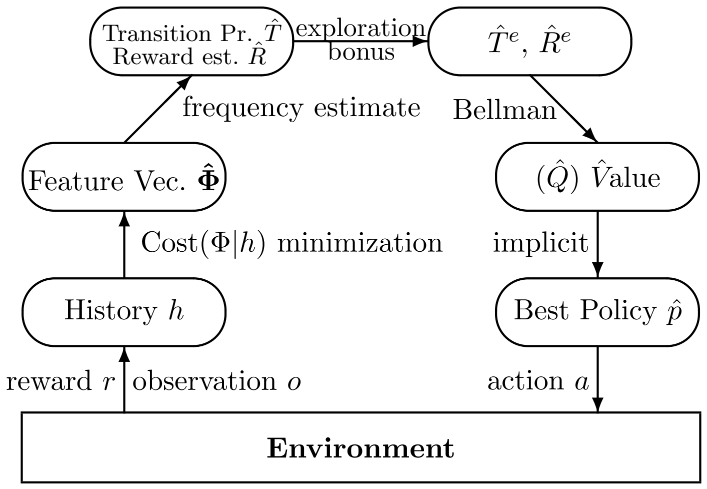
In practice, M can and should be chosen smaller like done in the original Cost function, and/or by using the restrictive model (8) for R, and/or by considering only non-zero frequencies (2). Analogous to (7) we seek a Φ that minimizes ICost().

Since action evaluation is based on (discounted) reward sums, not individual rewards, one may think of marginalizing $\mathrm{P}_U(\boldsymbol{r}|\boldsymbol{a},\Phi)$ even further, or coding rewards only approximately. Unfortunately, the algorithms in Section 6 that learn, explore, and exploit MDPs require knowledge of the (exact) individual rewards, so this improvement is not feasible.

8. Discussion

This section summarizes ΦMDP, relates it to previous work, and hints at more efficient incremental implementations and more realistic *structured* MDPs (dynamic Bayesian networks).

Summary. Learning from rewards in general environments is an immensely complex problem. In this paper I have developed a generic reinforcement learning algorithm based on sound principles. The key idea was to reduce general learning problems to finite state MDPs for which efficient learning, exploration, and exploitation algorithms exist. For this purpose I have developed a formal criterion for evaluating and selecting good "feature" maps Φ from histories to states. One crucial property of ΦMDP is that it neither requires nor learns a model of the complete observation space, but only for the reward-relevant observations as summarized in the states. The developed criterion has been inspired by MDL, which recommends to select the (coding) model that minimizes the length of a suitable code for the data at hand plus the complexity of the model itself. The novel and tricky part in ΦMDP was to deal with the states, since they are not bare observations, but model-dependent processed data. An improved Bayesian criterion, which integrates out the states, has also been derived. Finally, I presented a complete feature reinforcement learning algorithm ΦMDP-Agent(). The building blocks and computational flow are depicted in the following diagram:



Relation to previous work. As already indicated here and there, ΦMDP can be regarded as extending the frontier of many previous important approaches to RL and beyond: *Partially Observable MDPs (POMDPs)* are a very important generalization of MDPs (Kaelbling, Littman, and Cassandra, 1998). Nature is still assumed to be an MDP, but the states of nature are only partially observed via some non-injective or probabilistic function. Even for finite state space and known observation and transition functions, finding and even only approximating the optimal action is (harder than NP) hard (Lusena, Goldsmith, and Mundhenk, 2001; Madani, Hanks, and Condon, 2003). Lifting any of the assumptions causes conceptual problems, and when lifting more than one we enter scientific terra nullius. Assume a POMDP environment: POMDPs can formally (but not yet practically) be reduced to MDPs over so-called (continuous) belief states. Since ΦMDP reduces every problem to an MDP, it is conceivable that it reduces the POMDP to (an approximation of) its belief MDP. This would be a profound relation between ΦMDP and POMDP, likely leading to valuable insights into ΦMDP and proper algorithms for learning POMDPs. It may also help us to restrict the space of potentially interesting features Φ. *Predictive State Representations (PSRs)* are very interesting, but to this date in an even less developed stage (Singh et al., 2003) than POMDPs. *Universal AI* (Hutter, 2005) is able to optimally deal with arbitrary environments, but the resulting AIXI agent is computationally intractable (Hutter, 2007) and hard to approximate (Pankov, 2008; Poland and Hutter, 2006). *Bayesian RL* algorithms (Dearden, Friedman, and Andre, 1999; Duff, 2002; Poupart et al., 2006; Ross and Pineau, 2008) (see also Kumar and Varaiya, 1986, Chp.11) can be regarded as implementations of the AIξ models (Poland and Hutter, 2006), which are down-scaled versions of AIXI, but the enormous computational demand still severely limits this approach. ΦMDP essentially differs from "generative" Bayesian RL and AIξ in that it neither requires to specify nor to learn a (complete stochastic) observation model. It is a more "discriminative" approach (Liang and Jordan, 2008). Since ΦMDP "automatically" models only the relevant aspects of the environment, it should be computationally less demanding than full Bayesian RL. *State aggregation* methods have been suggested earlier for solving large-scale MDP planning problems by grouping (partitioning) similar states together, resulting in (much) smaller block MDPs (Givan, Dean, and Greig, 2003). But the bi-simulation metrics used require knowledge of the MDP transition probabilities. ΦMDP might be regarded as an approach that lifts this assumption. *Suffix trees* (McCallum, 1996) are a simple class of features Φ. ΦMDP combined with a local search function that expands and deletes leaf nodes is closely related to the U-Tree algorithm of McCallum (1996), with a related but likely different split&merge criterion. *Miscellaneous*: ΦMDP also extends the theory of model selection (e.g. MDL, Grünwald, 2007) from passive to active learning.

Incremental updates. As discussed in Section 5, most search algorithms are local in the sense that they produce a chain of "slightly" modified candidate solutions, here Φ. This suggests a potential speedup by computing quantities of interest incrementally, which becomes even more important in the ΦDBN case (Hutter, 2009a,c).

Computing Cost(Φ) takes at most time $O(|\mathcal{S}|^2|\mathcal{A}||\mathcal{R}|)$. If we split or merge two states, we can incrementally update the cost in time $O(|\mathcal{S}||\mathcal{A}||\mathcal{R}|)$, rather than computing it again from scratch. In practice, many transition $T_{ss'}^a$ don't occur, and Cost(Φ) can actually be computed much faster in time $O(|\{n_{ss'}^{ar}>0\}|)$, and incrementally even faster.

Iteration algorithms for (9) need an initial value for V or Q. We can take the estimate $\hat{V}$ from a previous Φ as an initial value for the new Φ. For a merge operation we can average the value of both states, for a split operation we could give them the same initial value. A significant further speedup can be obtained by using prioritized iteration algorithms that concentrate their time on badly estimated states, which are in our case (states close to) the new ones (Sutton and Barto, 1998).

Similarly, results from cycle n can be (re)used for the next cycle $n+1$. For instance, $\hat{V}$ can simply be reused as an initial value in the Bellman equations, and ICost(Φ) can be updated in time $O(|\mathcal{S}|^2)$ or even faster if U is sparse.

Feature dynamic Bayesian networks. The use of "unstructured" MDPs, even our Φ-optimal ones, is clearly limited to very simple tasks. Real world problems are structured and can often be represented by dynamic Bayesian networks (DBNs) with a reasonable number of nodes. Our Φ selection principle can be adapted from MDPs to the conceptually much more complex DBN case. The primary purpose of this Part I was to explain the key concepts on an as simple model as possible, namely unstructured finite MDPs, to set the stage for developing the more realistic ΦDBN in Part II (Hutter, 2009c).

Outlook. The major open problems are to develop smart Φ generation and smart stochastic search algorithms for Φ^{best}, and to determine whether minimizing (14) is the right criterion.

Acknowledgements. My thanks go to Pedro Ortega, Sergey Pankov, Scott Sanner, Jürgen Schmidhuber, and Hanna Suominen for feedback on earlier drafts.

References

Aarts, E. H. L., and Lenstra, J. K., eds. 1997. *Local Search in Combinatorial Optimization.* Discrete Mathematics and Optimization. Chichester, England: Wiley-Interscience.

Banzhaff, W.; Nordin, P.; Keller, E.; and Francone, F. 1998. *Genetic Programming.* San Francisco, CA, U.S.A.: Morgan-Kaufmann.

Barron, A. R. 1985. *Logically Smooth Density Estimation.* Ph.D. Dissertation, Stanford University.

Berry, D. A., and Fristedt, B. 1985. *Bandit Problems: Sequential Allocation of Experiments.* London: Chapman and Hall.

Brafman, R. I., and Tennenholtz, M. 2002. R-max – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research* 3:213–231.

Cover, T. M., and Thomas, J. A. 2006. *Elements of Information Theory.* Wiley-Intersience, 2nd edition.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based Bayesian Exploration. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, 150–159.

Duff, M. 2002. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes.* Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Amherst.

Dzeroski, S.; de Raedt, L.; and Driessens, K. 2001. Relational Reinforcement Learning. *Machine Learning* 43:7–52.

Fishman, G. 2003. *Monte Carlo.* Springer.

Givan, R.; Dean, T.; and Greig, M. 2003. Equivalence Notions and Model Minimization in Markov Decision Processes. *Artificial Intelligence* 147(1–2):163–223.

Goertzel, B., and Pennachin, C., eds. 2007. *Artificial General Intelligence.* Springer.

Gordon, G. 1999. *Approximate Solutions to Markov Decision Processes.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Grünwald, P. D. 2007. *The Minimum Description Length Principle.* Cambridge: The MIT Press.

Guyon, I., and Elisseeff, A., eds. 2003. *Variable and Feature Selection.* JMLR Special Issue: MIT Press.

Hastie, T.; Tibshirani, R.; and Friedman, J. H. 2001. *The Elements of Statistical Learning.* Springer.

Hutter, M. 2005. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Berlin: Springer. 300 pages, http://www.hutter1.net/ai/uaibook.htm.

Hutter, M. 2007. Universal Algorithmic Intelligence: A Mathematical Top→Down Approach. In *Artificial General Intelligence.* Berlin: Springer. 227–290.

Hutter, M. 2009a. Feature Dynamic Bayesian Networks. In *Proc. 2nd Conf. on Artificial General Intelligence (AGI'09)*, volume 8, 67–73. Atlantis Press.

Hutter, M. 2009b. Feature Markov Decision Processes. In *Proc. 2nd Conf. on Artificial General Intelligence (AGI'09)*, volume 8, 61–66. Atlantis Press.

Hutter, M. 2009c. Feature Reinforcement Learning: Part II: Structured MDPs. *In progress. Will extend Hutter (2009a).*

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101:99–134.

Kearns, M. J., and Singh, S. 1998. Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, 260–268. Morgan Kaufmann, San Francisco, CA.

Koza, J. R. 1992. *Genetic Programming.* The MIT Press.

Kumar, P. R., and Varaiya, P. P. 1986. *Stochastic Systems: Estimation, Identification, and Adaptive Control.* Englewood Cliffs, NJ: Prentice Hall.

Legg, S., and Hutter, M. 2007. Universal Intelligence: A Definition of Machine Intelligence. *Minds & Machines* 17(4):391–444.

Legg, S. 2008. *Machine Super Intelligence.* Ph.D. Dissertation, IDSIA, Lugano.

Li, M., and Vitányi, P. M. B. 2008. *An Introduction to Kolmogorov Complexity and its Applications.* Berlin: Springer, 3rd edition.

Liang, P., and Jordan, M. 2008. An Asymptotic Analysis of Generative, Discriminative, and Pseudolikelihood Estimators. In *Proc. 25th International Conf. on Machine Learning (ICML'08)*, volume 307, 584–591. ACM.

Liu, J. S. 2002. *Monte Carlo Strategies in Scientific Computing.* Springer.

Lusena, C.; Goldsmith, J.; and Mundhenk, M. 2001. Nonapproximability Results for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research* 14:83–103.

MacKay, D. J. C. 2003. *Information theory, inference and learning algorithms.* Cambridge, MA: Cambridge University Press.

Madani, O.; Hanks, S.; and Condon, A. 2003. On the Undecidability of Probabilistic Planning and Related Stochastic Optimization Problems. *Artificial Intelligence* 147:5–34.

McCallum, A. K. 1996. *Reinforcement Learning with Selective Perception and Hidden State.* Ph.D. Dissertation, Department of Computer Science, University of Rochester.

Ng, A. Y.; Coates, A.; Diel, M.; Ganapathi, V.; Schulte, J.; Tse, B.; Berger, E.; and Liang, E. 2004. Autonomous Inverted Helicopter Flight via Reinforcement Learning. In *ISER*, volume 21 of *Springer Tracts in Advanced Robotics*, 363–372. Springer.

Pankov, S. 2008. A Computational Approximation to the AIXI Model. In *Proc. 1st Conference on Artificial General Intelligence*, volume 171, 256–267.

Pearlmutter, B. A. 1989. Learning State Space Trajectories in Recurrent Neural Networks. *Neural Computation* 1(2):263–269.

Poland, J., and Hutter, M. 2006. Universal Learning of Repeated Matrix Games. In *Proc. 15th Annual Machine Learning Conf. of Belgium and The Netherlands (Benelearn'06)*, 7–14.

Poupart, P.; Vlassis, N. A.; Hoey, J.; and Regan, K. 2006. An Analytic Solution to Discrete Bayesian Reinforcement Learning. In *Proc. 23rd International Conf. on Machine Learning (ICML'06)*, volume 148, 697–704. Pittsburgh, PA: ACM.

Puterman, M. L. 1994. *Markov Decision Processes — Discrete Stochastic Dynamic Programming.* New York, NY: Wiley.

Raedt, L. D.; Hammer, B.; Hitzler, P.; and Maass, W., eds. 2008. *Recurrent Neural Networks - Models, Capacities, and Applications*, volume 08041 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany.

Ring, M. 1994. *Continual Learning in Reinforcement Environments.* Ph.D. Dissertation, University of Texas, Austin.

Ross, S., and Pineau, J. 2008. Model-Based Bayesian Reinforcement Learning in Large Structured Domains. In *Proc. 24th Conference in Uncertainty in Artificial Intelligence (UAI'08)*, 476–483. Helsinki: AUAI Press.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research* 2008(32):663–704.

Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence. A Modern Approach.* Englewood Cliffs, NJ: Prentice-Hall, 2nd edition.

Sanner, S., and Boutilier, C. 2009. Practical Solution Techniques for First-Order MDPs. *Artificial Intelligence* 173(5–6):748–788.

Schmidhuber, J. 2004. Optimal Ordered Problem Solver. *Machine Learning* 54(3):211–254.

Schwarz, G. 1978. Estimating the Dimension of a Model. *Annals of Statistics* 6(2):461–464.

Singh, S.; Littman, M.; Jong, N.; Pardoe, D.; and Stone, P. 2003. Learning Predictive State Representations. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, 712–719.

Strehl, A. L.; Diuk, C.; and Littman, M. L. 2007. Efficient Structure Learning in Factored-State MDPs. In *Proc. 27th AAAI Conference on Artificial Intelligence*, 645–650. Vancouver, BC: AAAI Press.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Szita, I., and Lörincz, A. 2008. The Many Faces of Optimism: a Unifying Approach. In *Proc. 12th International Conference (ICML 2008)*, volume 307.

Wallace, C. S. 2005. *Statistical and Inductive Inference by Minimum Message Length.* Berlin: Springer.

Willems, F. M. J.; Shtarkov, Y. M.; and Tjalkens, T. J. 1997. Reflections on the Prize Paper: The Context-Tree Weighting Method: Basic Properties. *IEEE Information Theory Society Newsletter* 20–27.

Wolpert, D. H., and Macready, W. G. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.

Appendix A. List of Notation

Interface structures

$\mathcal{O}$	= finite or infinite set of possible observations
$\mathcal{R}$	= $\{0,1\}$ or $[0,R_{max}]$ or other set of rewards
$\mathcal{A}$	= (small) finite set of actions
$n \in I\!N$	= current time
$o_t r_t a_t$	= $ora_t \in \mathcal{O}\times\mathcal{R}\times\mathcal{A}$ = true observation, reward, action at time t

Internal structures for ΦMDP

log	= binary logarithm
$t\in\{1,...,n\}$	= any time
$i\in\{1,...,m\}$	= any state index
$\boldsymbol{x}=x_{1:n}$	= $x_1...x_n$ (any x)
$x_+,x_*,\boldsymbol{x}_\bullet$	= $\sum_j x_j$, $\bigcup_j x_j$, $(x_1,...,x_l)$ (any x,j,l)
$\hat{X}$	= estimate of X (any X)
$\mathcal{H}$	= $(\mathcal{O}\times\mathcal{R}\times\mathcal{A})^*\times\mathcal{O}\times\mathcal{R}$ = possible histories
h_n	= $ora_{1:n-1}o_n r_n$ = actual history at time n
$\mathcal{S}$	= $\{s^1,...,s^m\}$ = internal finite state space (can vary with n)
$\Phi:\mathcal{H}\to\mathcal{S}$	= state or feature summary of history
s_t	= $\Phi(h_t)\in\mathcal{S}$ = realized state at time t
$\mathrm{P}(\cdot)$	= probability over states and rewards or parts thereof
$\mathrm{CL}(\cdot)$	= code length
MDP	= $(\mathcal{S},\mathcal{A},T,R)$ = Markov Decision Process
$T^a_{ss'}$	= $\mathrm{P}(s_t=s'\vert s_{t-1}=s,a_{t-1}=a)$ = transition matrix
$s\xrightarrow{a}s'(r')$	= action a in state s resulted in state s' (and reward r')
$\mathcal{T}^{ar'}_{ss'}$	= set of times $t\in\{1,...,n\}$ at which $s\xrightarrow{a}s'r'$
$n^{ar'}_{ss'}$	= $\vert\mathcal{T}^{ar'}_{ss'}\vert$ = number of times $t\in\{1,...,n\}$ at which $s\xrightarrow{a}s'r'$
$\mathrm{Cost}(\Phi\vert h)$	= cost (evaluation function) of Φ based on history h
$\mathrm{ICost}(\Phi\vert h)$	= improved cost function
Q^{*a}_s,V^*_s	= optimal (Q) Value (of action a) in state s
$\gamma\in[0;1)$	= discount factor ($(1-\gamma)^{-1}$ is effective horizon)