Hierarchical Reinforcement Learning

Susanne Schlötzer

Institute of Automatic Control Engineering
Technische Universität München

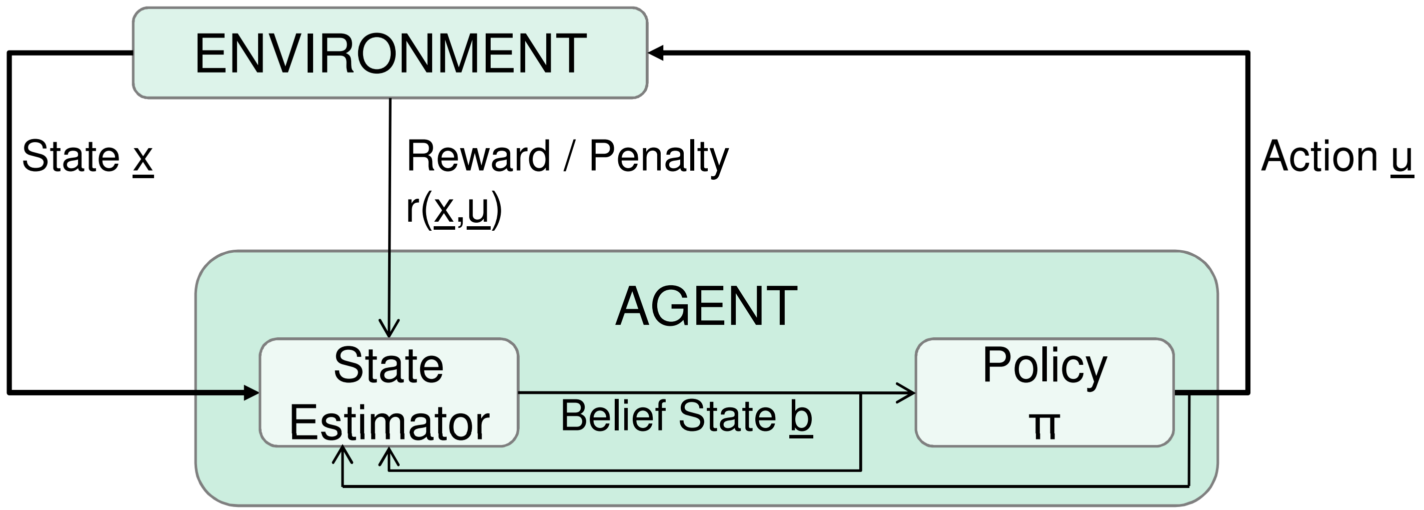
susanne.schloetzer@mytum.de

Content

- Introduction to Reinforcement Learning (RL)
- Motivation for hierarchical architectures
- Basic scheme of hierarchical RL
- Task decomposition by sub-goals
  - Stand-up task of a robot
  - Learning in the upper-level
  - Learning in the lower-level
  - Evaluation
- Further hierarchical approaches in RL
- Conclusion

Introduction to Reinforcement Learning (I)

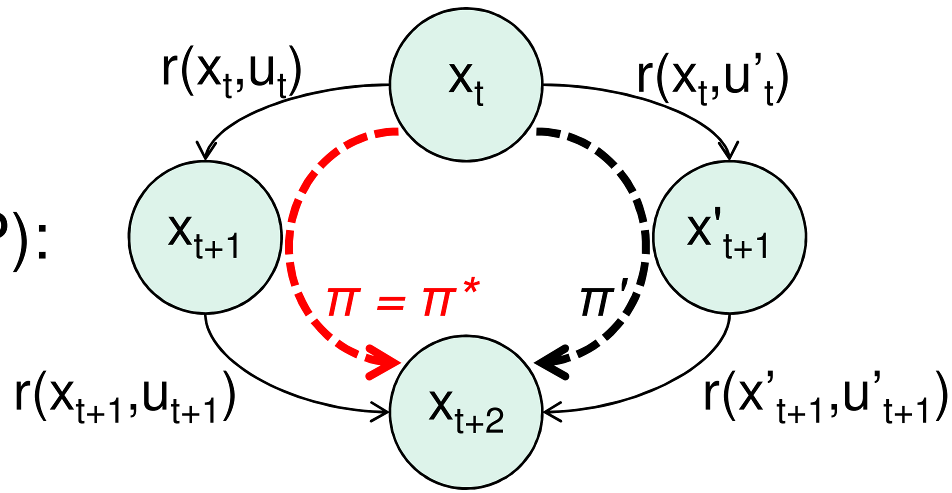
- Parts of a RL problem
  1) (Dynamic) Environment
  2) Reward function: *r*
  3) Value function: $V(\underline{x})$



- "Learning with a critic" (reinforcement learning) in contrast to "learning with a teacher" (supervised learning)

Introduction to Reinforcement Learning (II)

- Objective
  - Learning of the best policy to solve a problem by trial-and-error runs
  - The best policy (π^*) is given by finding a sequence of actions that maximizes the expected cumulative reward $V(\underline{x}_t)$:
    - $V^{\pi}(\underline{x}_t) = \mathrm{E}\left[\sum_{i=1}^{\infty} \gamma^{i-1} \cdot r_{t+i}\right]$
    - $V^{*}(\underline{x}_t) = \max_{\pi} V^{\pi}(\underline{x}_t) \quad , \quad \forall \underline{x}_t$
- Example of a deterministic Markov Decision Process (MDP):



Motivation for Hierarchical Architectures

- Apply RL to real-world problems, e.g. complex machine learning systems
  - RL methods scale badly with the size of state spaces
  - RL methods scale badly with the size of action spaces
  - RL methods scale badly with the length of policies

- Use hierarchical RL for systems where
  - the duration of the learning period matters
  - Online learning is required
    - Online planning
    - Online adaptation to the environment

Basic Scheme of Hierarchical RL

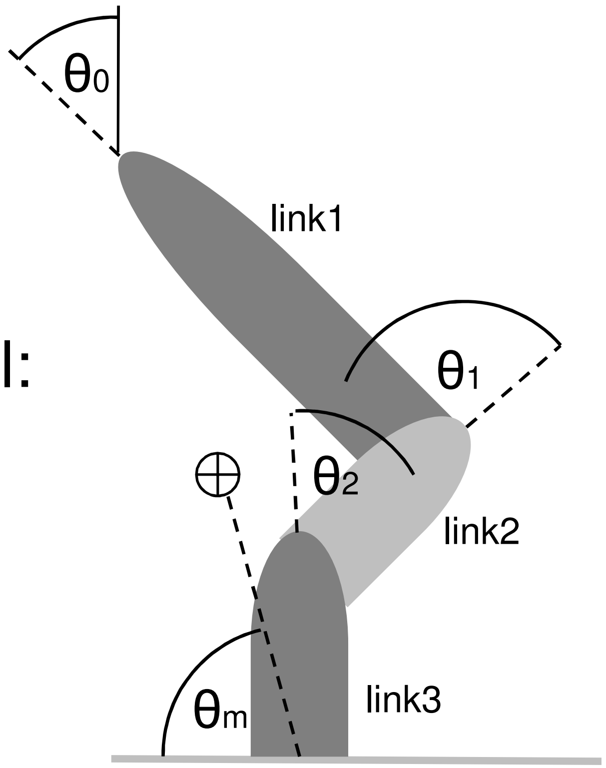
- Hierarchical architecture of RL refers to
  - Multiple levels and/or
  - Multiple temporal scales and/or
  - Multiple spatial scales

- Construction of higher levels
  - Macro-actions
  - Sub-goals
  - Multiple time scales

- Frequently based on Semi Markov Decision Processes (SMDP)

Example: Stand-up Task of a Robot (I)

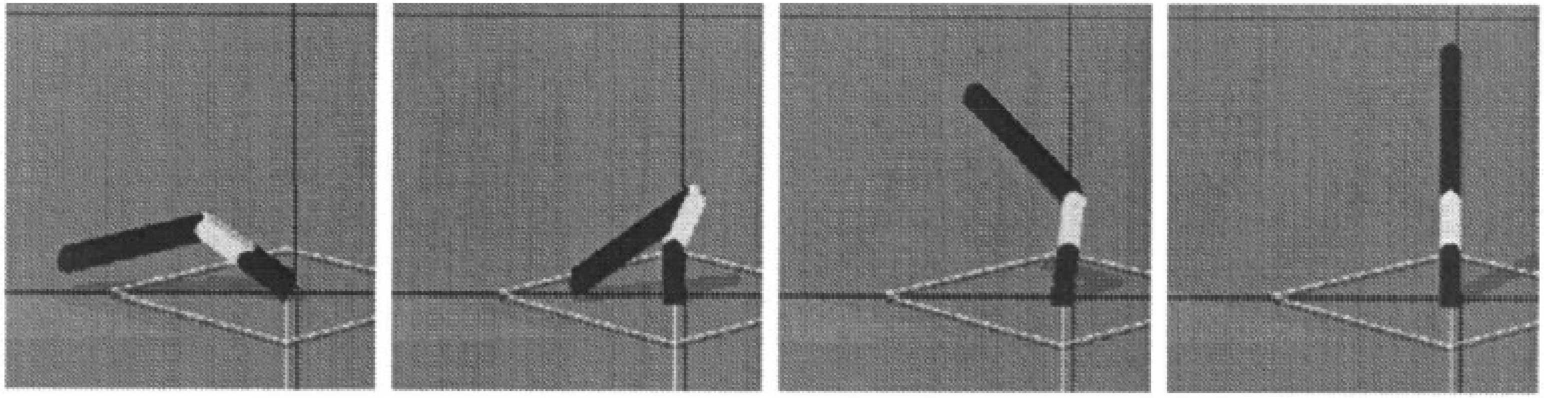
- Task decomposition by sub-goals as proposed by J. Morimoto and K. Doya
  - Upper level: exploration of a low-dimensional state space
  - Lower level: optimization of local trajectories in the high-dimensional state space



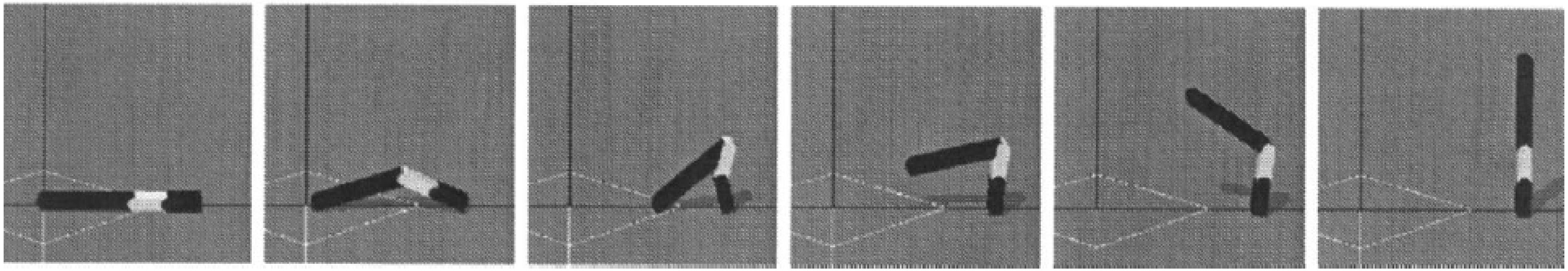
- Two-joint, three-link robot
  - State variables chosen for the upper level: $\underline{X}(T) = (\theta_m, \theta_1, \theta_2)$
  - State variables used in the lower level: $\underline{x}(t) = (\theta_0, \theta_1, \theta_2, \dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2)$

Example: Stand-up Task of a Robot (II)

- Episode of a successful stand-up trial
  - Upper level

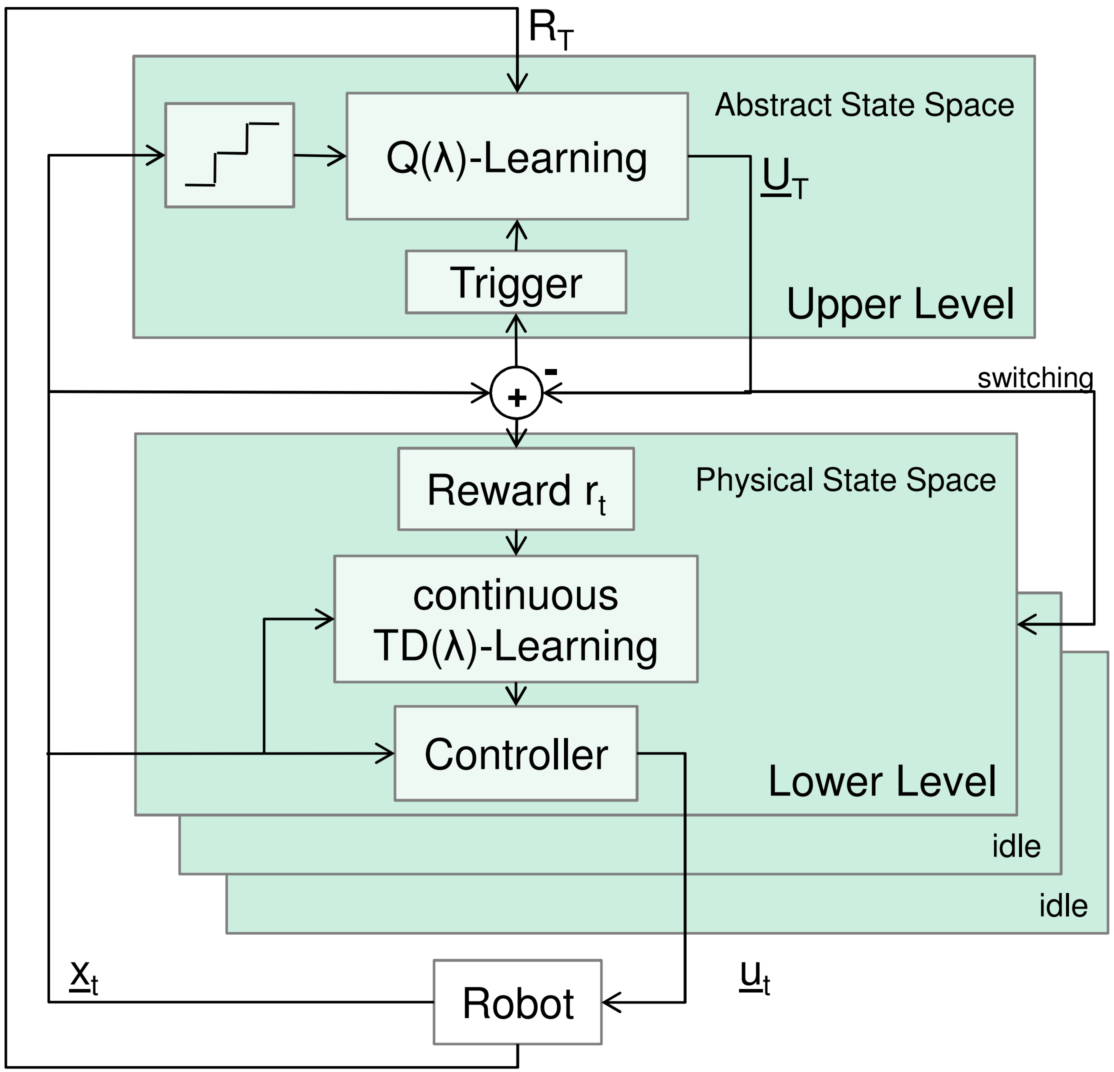


  - Lower level



Example: Stand-up Task of a Robot (III)

- Hierarchical architecture



- Global exploration of sub-goals
- $R_T = R_{main} + R_{sub}$

$$R_{main} = \begin{cases} 1 & \text{, on success of stand-up} \\ 0 & \text{, on failure} \end{cases}$$

$$R_{sub} = \begin{cases} 1 & \text{, final goal achieved} \\ 0.25 \cdot \left(\frac{Y}{L} + 1 \right) & \text{, sub-goal achieved} \\ 0 & \text{, on failure} \end{cases}$$

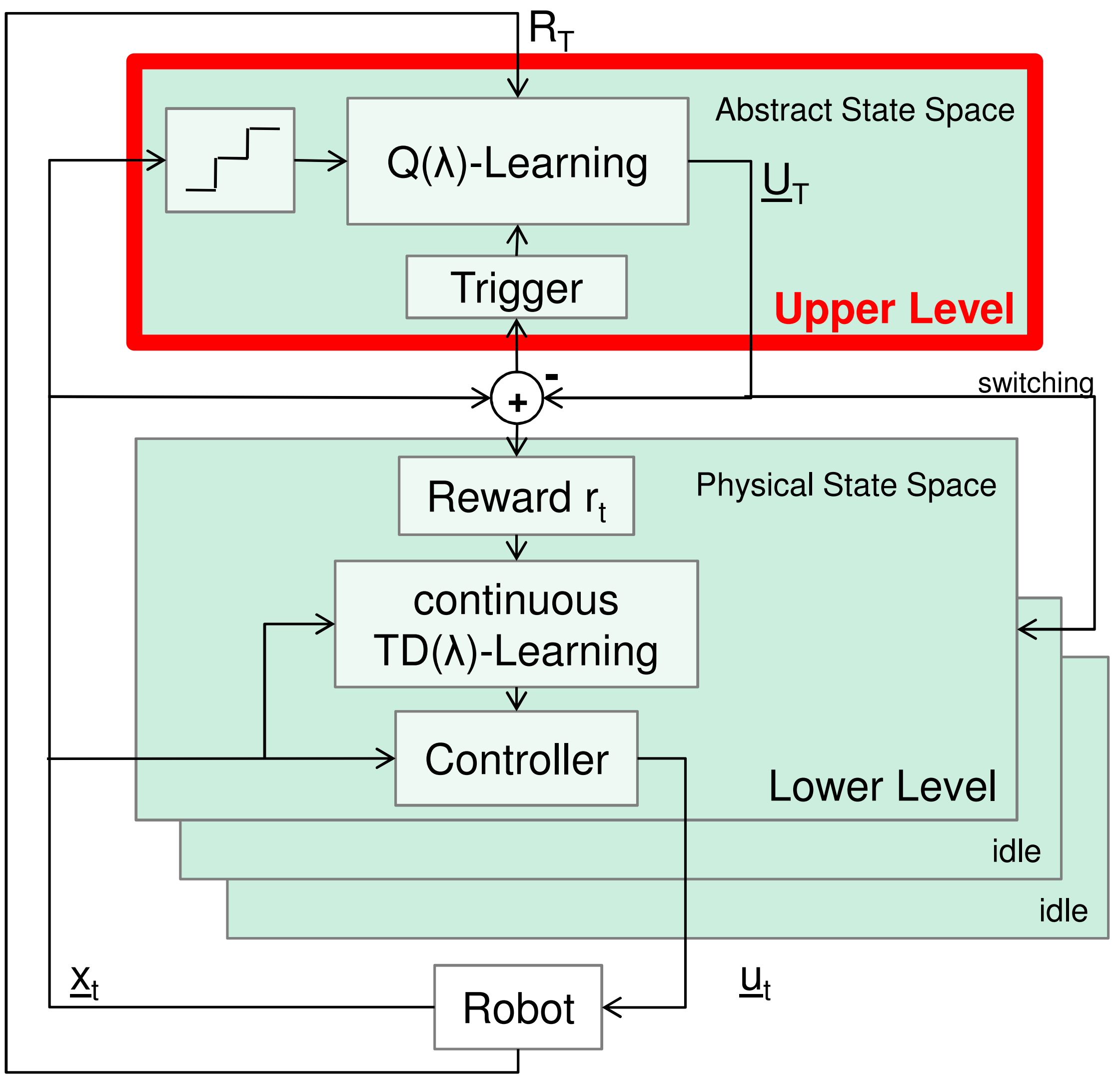
- Optimization of local trajectories

$$r(\underline{\theta}, \underline{\tilde{\theta}}) = \exp\left(- \frac{\left\| \underline{\theta} - \underline{\tilde{\theta}} \right\|^2}{s_\theta^2} \right) - 1 \quad \text{, during control}$$

$$r(t) = \begin{cases} \exp\left(- \frac{\left\| \underline{\dot{\theta}} - \underline{\dot{\tilde{\theta}}} \right\|^2}{s_{\dot{\theta}}^2} \right) & \text{, sub-goal achieved} \\ -1.5 & \text{, fall down} \end{cases}$$

Upper Level Learning

- Learning of a discrete sequence of sub-goals



Choice of Actions in the Upper Level

- No perfect knowledge of the environment
  - Exploration strategy is required to query the model
  - Smooth transition from exploration to exploitation
  - Probabilistic choice of an action $\underline{u}_t$ at time t (Boltzmann distribution):

$$P(\underline{u}_t \mid \underline{x}_t) = \frac{\mathrm{E}\left[\beta \cdot Q(\underline{x}_t, \underline{u}_t)\right]}{\sum_{\underline{b} \in \mathrm{A}(\underline{x})} \mathrm{E}\left[\beta \cdot Q(\underline{x}_t, \underline{b})\right]}$$

  - With:
    - β: exploration factor (inverse temperature)
    - $A(\underline{x})$: set of possible actions at state $\underline{x}$

Q(λ)-Learning (I)

- Used for (general) Markov Decision Processes (MDP)
  - Set of potential successor states for a given action in a given state
  - Value iteration is not applicable in practise
- Value function $V(\underline{x}_t)$ is replaced by <u>Q-function</u> $Q(\underline{x}_t,\underline{u}_t)$
  - Q-function is frequently used in the context of control
  - The state value $V^*(\underline{x}_t)$ and the expected accumulated reward $Q^*(\underline{x}_t,\underline{u}_t)$ of an action $\underline{u}_t$ taken in state $\underline{x}_t$ are linked by:

$$V^*(\underline{x}_t) = \max_{\underline{u}_t} Q^*(\underline{x}_t,\underline{u}_t) = \max_{\underline{u}_t} \mathrm{E}\left[\sum_{i=1}^{\infty} \gamma^{i-1} \cdot r_{t+1}\right] = \max_{\underline{u}_t} \mathrm{E}\left[r_{t+1} + \gamma \cdot V^*(\underline{x}_{t+1})\right]$$

    - $0 \le \gamma < 1$: discount factor to keep future rewards discounted for infinite horizon models

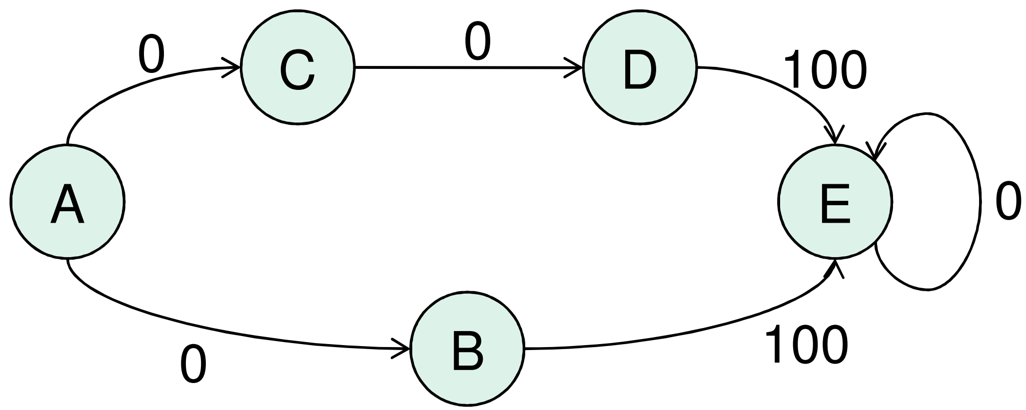
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	0	0	–	–
B	–	–	–	–	0
C	–	–	–	0	–
D	–	–	–	–	0
E	–	–	–	–	0



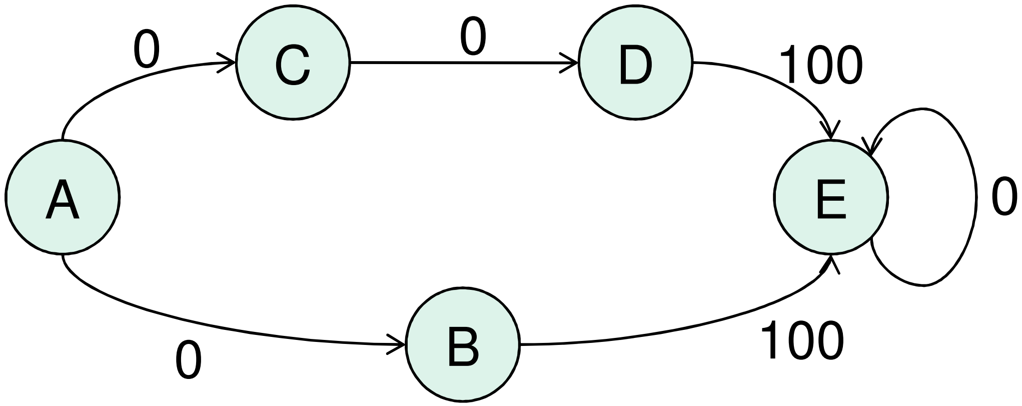
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: $\alpha=0.1$, $\gamma=0.9$

State/Action	A	B	C	D	E
A	–	0	0	–	–
B	–	–	–	–	10
C	–	–	–	0	–
D	–	–	–	–	0
E	–	–	–	–	0



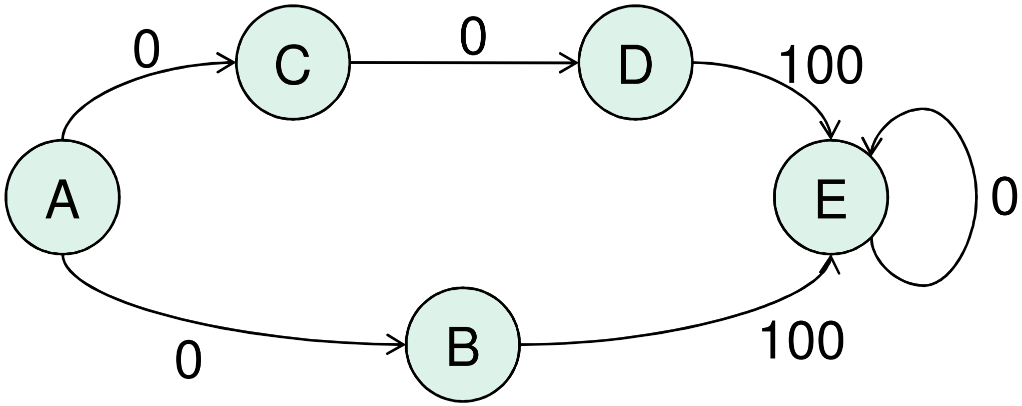
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	0.9	0	–	–
B	–	–	–	–	10
C	–	–	–	0	–
D	–	–	–	–	0
E	–	–	–	–	0



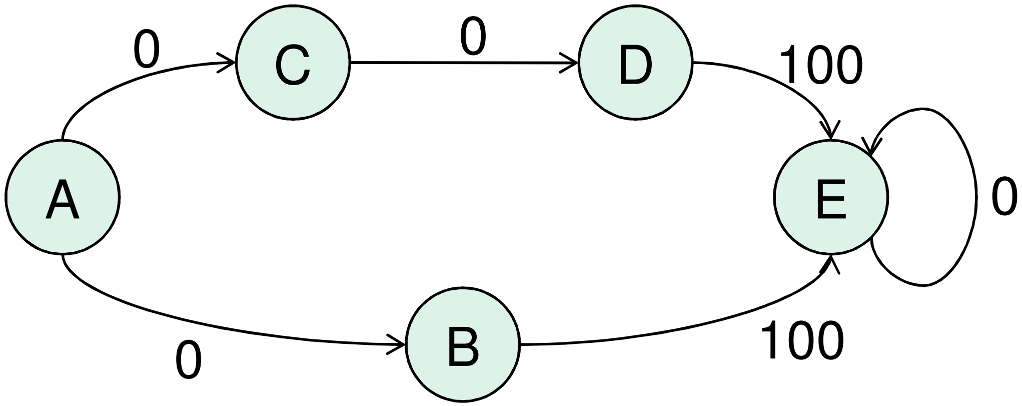
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	*A*	*B*	*C*	*D*	*E*
A	–	0.9	0	–	–
B	–	–	–	–	19
C	–	–	–	0	–
D	–	–	–	–	0
E	–	–	–	–	0



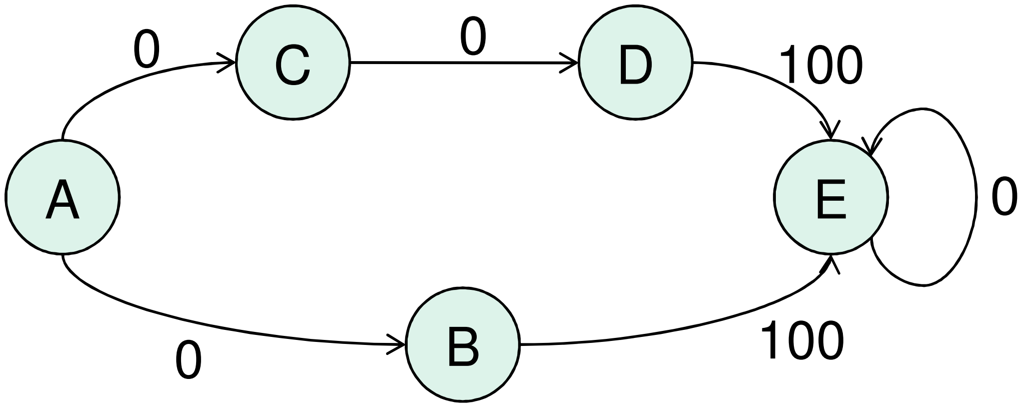
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	0.9	0	–	–
B	–	–	–	–	19
C	–	–	–	0	–
D	–	–	–	–	10
E	–	–	–	–	0



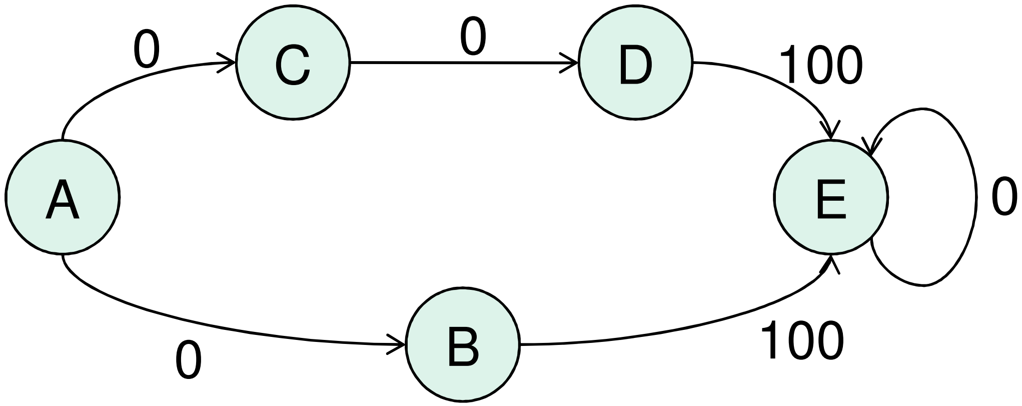
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	0.9	0	–	–
B	–	–	–	–	19
C	–	–	–	0.9	–
D	–	–	–	–	19
E	–	–	–	–	0



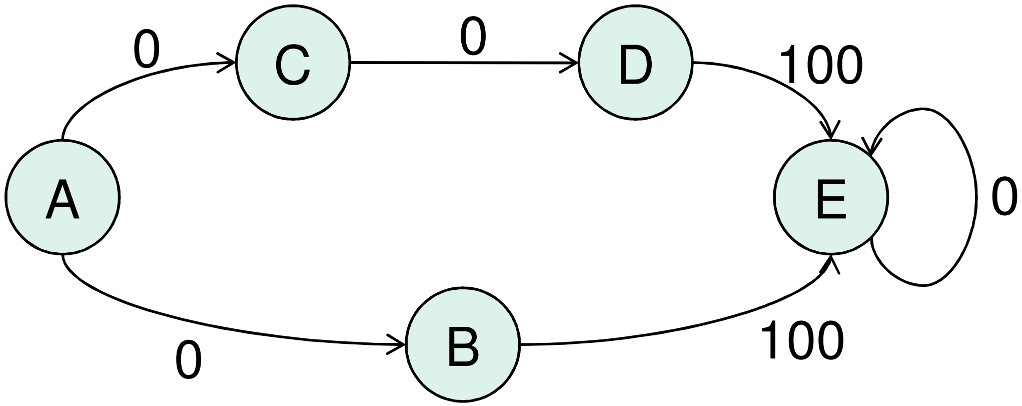
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	0.9	0.081	–	–
B	–	–	–	–	19
C	–	–	–	2.52	–
D	–	–	–	–	27
E	–	–	–	–	0



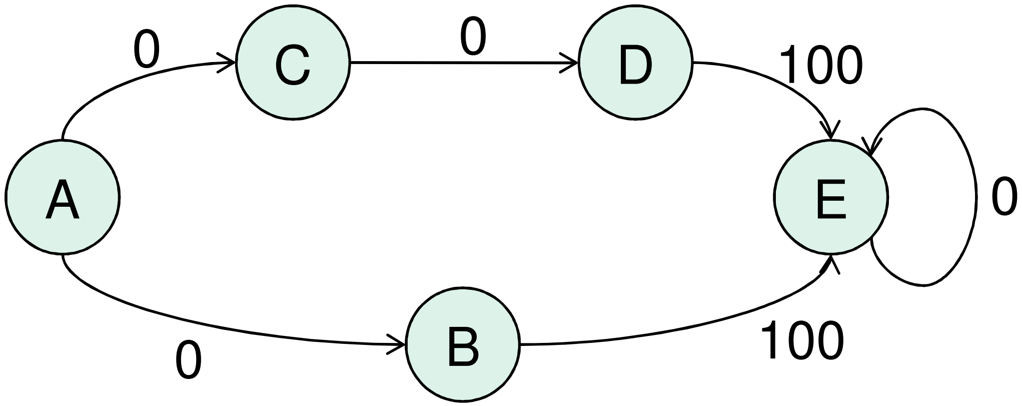
Q(λ)-Learning (II)

- Special case: Q(0)-learning, e.g. one-step Q-learning
- Update rule for Q(0)-learning with the learning rate α:

$$Q(\underline{x},\underline{u}) \leftarrow Q(\underline{x},\underline{u}) + \alpha \cdot \left[r + \gamma \cdot \max_{\underline{u}'} Q(\underline{x}',\underline{u}') - Q(\underline{x},\underline{u}) \right]$$

  – Example: α=0.1, γ=0.9

State/Action	A	B	C	D	E
A	–	90	81	–	–
B	–	–	–	–	100
C	–	–	–	90	–
D	–	–	–	–	100
E	–	–	–	–	0

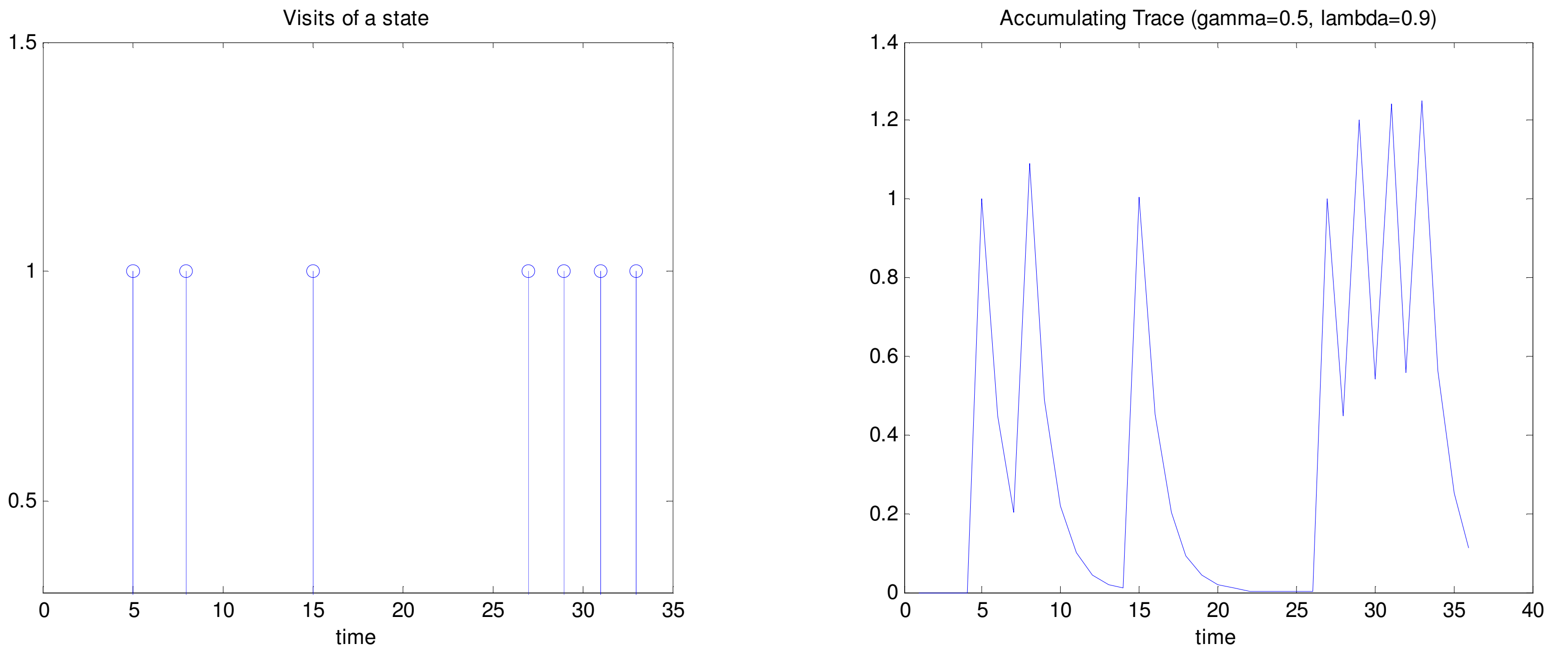


Q(λ)-Learning (III)

- λ≠0: Update the values of previously occurring visits
- Accumulating <u>eligibility trace</u> *e(<u>x</u>)*:

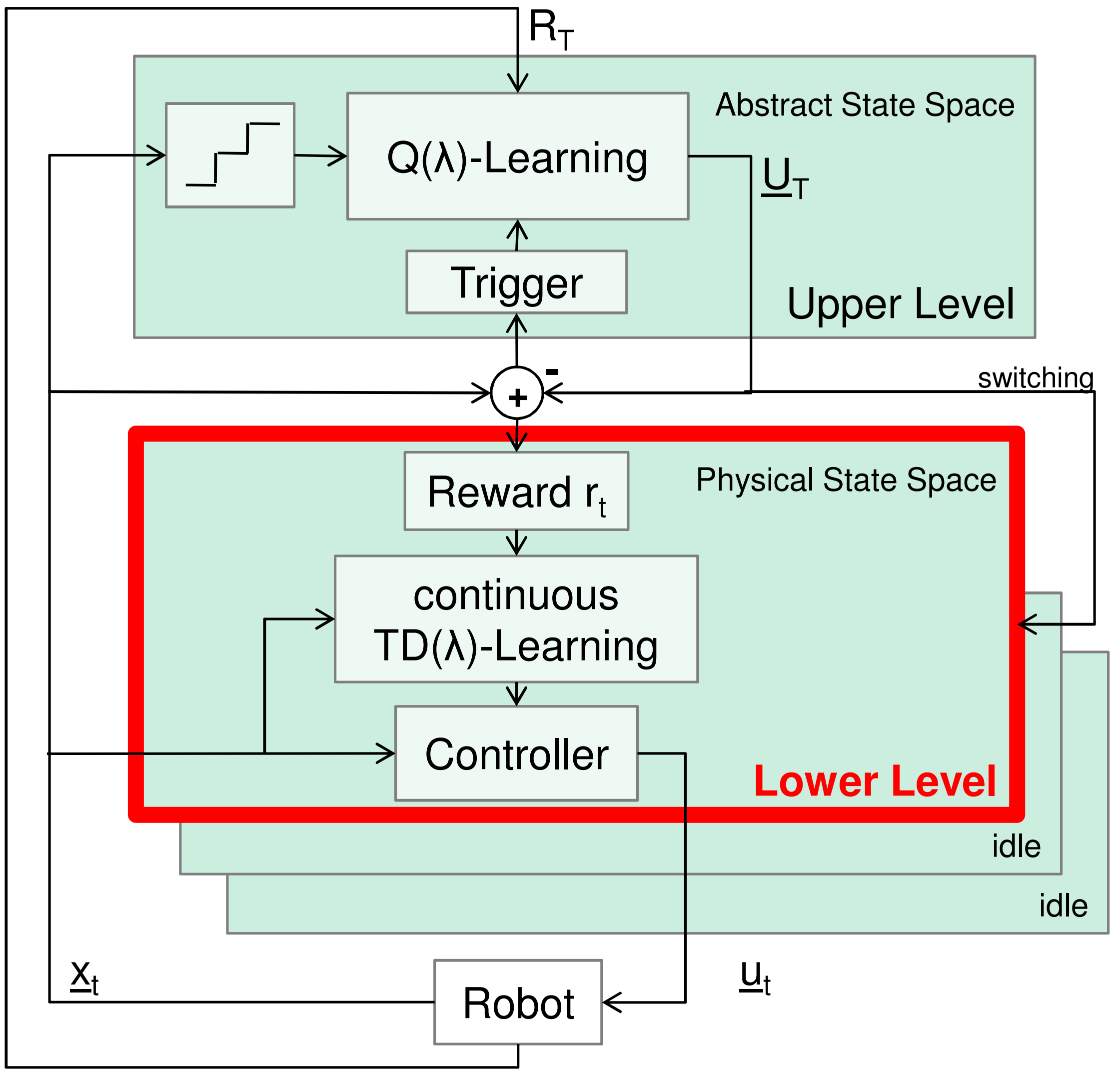
$$e_{t+1}(\underline{x}) = \begin{cases} \gamma \cdot \lambda \cdot e_t(\underline{x}) & \text{, if } \underline{x} \neq \underline{x}_t \\ \gamma \cdot \lambda \cdot e_t(\underline{x}) + 1 & \text{, if } \underline{x} = \underline{x}_t \end{cases}$$

  – 0≤λ≤1: trace decay parameter



Lower Level Learning

- Learning of local trajectories



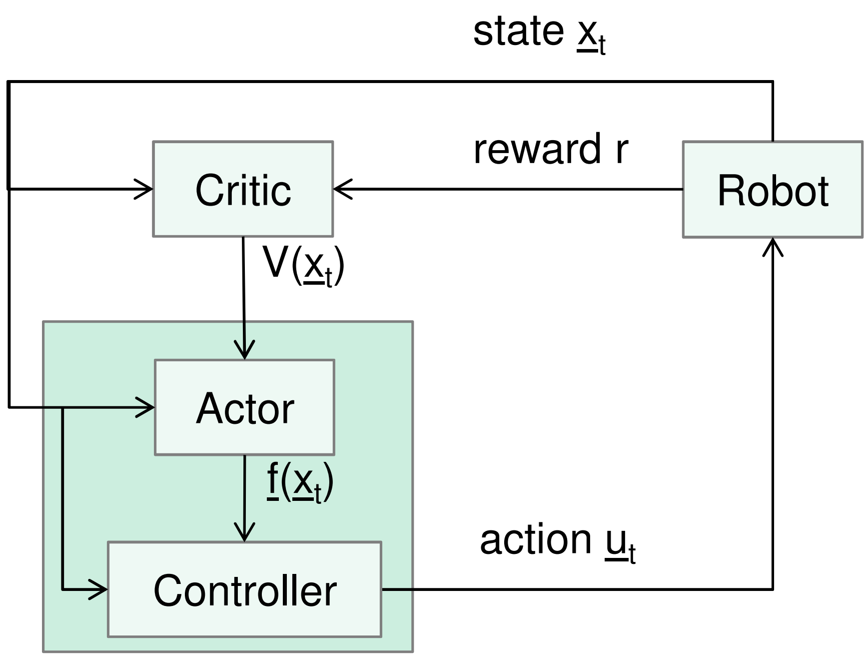
Choice of Actions in the Lower Level

- Continuous action-space in the lower level requires a different exploration strategy

- Usage of normalized Gaussian basis functions instead of a Boltzmann distribution in order to select actions
  - Continuous function approximator (here: learning of local trajectories)
  - Learning of non-linear functions (here: non-linear control function)
  - Frequently converges to local optima (here: global exploration of the state space in the upper level)

Continuous TD(λ)-Learning with Actor-Critic Method (I)

- Actor-critic method uses two function approximators:
  - Critic learns the state-value function that predicts the accumulated future reward $V(\underline{x}_t)$ at state $\underline{x}_t$
  - Actor learns the control functions $f(\underline{x}_t)$ that specify non-linear feedback control laws



Continuous TD(λ)-Learning with Actor-Critic Method (II)

- Update rule for continuous Temporal-Difference (TD) Learning
  - $\dot{w} = \alpha \cdot \delta_t \cdot g_t$
    - with the state-value prediction error

$$\delta_t = r_t - \frac{1}{\tau} \cdot V(\underline{x}_t) + \frac{dV(\underline{x}_t)}{dt}$$

      - δ_t: Hamiltonian, e.g. continuous equivalent to Bellman's residual
      - τ : time factor
    - α: learning rate
    - g_t:
      - Update of the actor: normalized Gaussian basis function weighted with a noise term for exploration
      - Update of the critic: eligibility trace of a basis function

Evaluation (I)

- Increase of the learning speed and the success rate of the approach successfully demonstrated in simulations

- Simulation results successfully transferred to a real robot that has to accomplish the stand-up task

- Task-specific optimization of parameters required
  - Choice of an appropriate subset of state variables for the upper level
  - Choice of an appropriate action step size
  - Choice of an appropriate reward function
  - Experimentation-sensitivity of trace-decay parameters

Evaluation (II)

- No formal convergence proof exists
  - Policy might be sub-optimal when combining the sub-problems

- A-priori information can be easily included

- Approaches providing reusability for several tasks
  - Refers to lower-level modules
  - Compositional Q-Learning [S. Singh]
  - Nested Q-Learning [B.L. Digney]

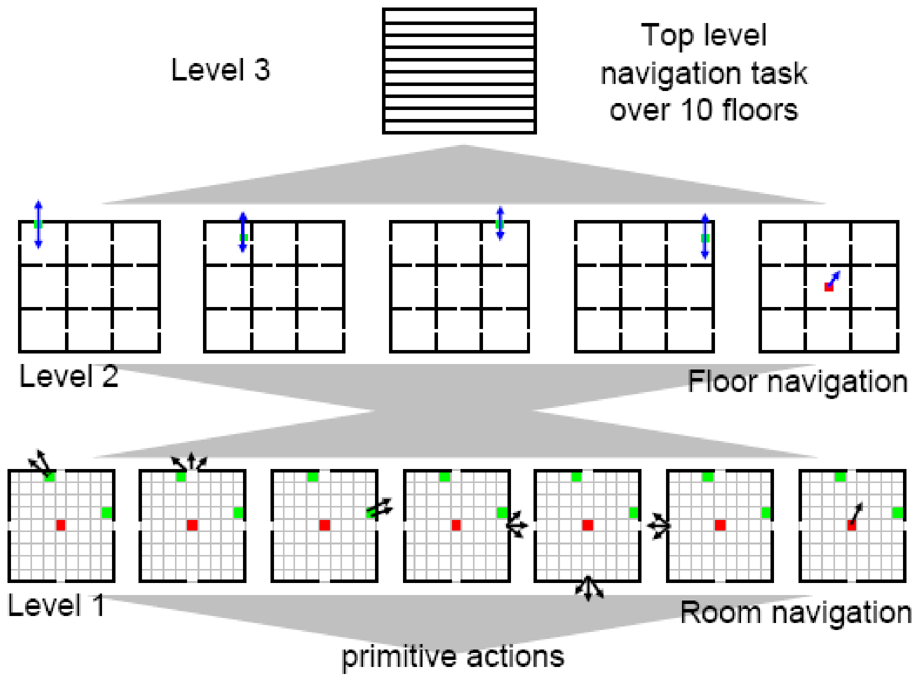
Further Approaches in hierarchical RL (I)

- Options-formalism [Sutton et al.]
  - Generalization of actions to include courses of actions
  - Execution of an option:
    - Policy π determines which actions are selected from the input set S
    - Option is terminated stochastically according to the termination condition β
- Hierarchies of Abstract Machines (HAMs) [Parr et al.]
  - Supervisor in the higher level that intervenes when its state enters a set of boundary states
  - Switching between several regulators in the lower level
- MAXQ framework [Dieterrich]
  - Decomposition of a MDP into a set of subtasks
    - Hierarchy of SMDPs whose solutions can be learned simultaneously
    - Hierarchical architecture can be represented in a task graph

Further Approaches in hierarchical RL (II)

- Dynamic Abstraction
  - Temporally-extended activities assess which variables have to be considered
  - Learning to set up task hierarchies automatically
  - Representative: HEXQ

    - Construction of a task hierarchy using HEXQ (after Hengst)



Conclusion

- High practical importance to real world problems
  - Reasonable learning speed
  - Can deal with high-dimensional state spaces
  - Provide reusability (some approaches)
- Various hierarchical approaches exist
  - Application-specific optimization of RL architectures
  - Convergence to optimal policy not always guaranteed
- Issue of recent research
  - Dynamic abstractions
  - Concurrent activities
  - Multi-agent strategies

References

- J. Morimoto, K. Doya. (2001). *Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning*. Elsevier Science, Robotics and Autonomous Systems, vol. 36, pp. 37-51.
- H. Miyamoto, J. Morimoto, K. Doya, M. Kawato. (2004). *Reinforcement learning with via-point representation*. Elsevier Science, Neural Networks, vol. 17, pp. 299-305.
- E. Alpaydin. (2004). *Introduction to Machine Learning*. MIT Press, Cambridge, Massachusetts.
- L.P. Kaelbling. (1996). *Recent Advances in Reinforcement Learning*. Kluwer academic publishers, Boston, Dordrecht, London.
- A.G. Barto, S. Mahadevan. (2003). *Recent Advances in Hierarchical Reinforcement Learning*. Kluwer Academic Publishers. Discrete Event Dynamic Systems: Theory and Application, vol. 13, pp. 41-77.