

# Learning a Transfer Function for Reinforcement Learning Problems

---

Tom Croonenborghs

*Biosciences and Technology Department, KH Kempen University College, Geel, Belgium*

Kurt Driessens, Maurice Bruynooghe

*Declarative Languages and Artificial Intelligence, Katholieke Universiteit Leuven, Leuven, Belgium*

K.U.Leuven Association



---

AAAI'08: Transfer Learning for Complex Tasks  
July 14 2008, Chicago



# Inductive Transfer for Reinforcement Learning

## Problem

Reinforcement Learning (RL) often approached as **tabula rasa** learning technique

- forced to perform **random exploration**
- quickly becomes infeasible or impractical in complex domains

## Solution: Use Transfer Learning!

- learn in a **source task**
- leverage knowledge in **target task**

A lot of different approaches, e.g. see Lisa's talk

- often a hand-coded mapping is needed to relate features from a source task to a target task



# Inductive Transfer for Reinforcement Learning

## Problem

Reinforcement Learning (RL) often approached as **tabula rasa** learning technique

- forced to perform **random exploration**
- quickly becomes infeasible or impractical in complex domains

## Solution: Use Transfer Learning!

- learn in a **source task**
- leverage knowledge in **target task**

A lot of different approaches, e.g. see Lisa's talk

- often a hand-coded mapping is needed to relate features from a source task to a target task



# How to get rid of the expert?

## Desiderata

Transfer knowledge from a source task to a target task with a different state (and action) representation **without** the need for a human-designed mapping

## Some Existing Work

**Liu & Stone 2006** Use a graph-matching algorithm to find similarities between state variables

- need for a complete and correct transition function

**Kuhlmann & Stone 2007** Similar approach tailored for GGP

**Taylor et al. 2007** Learn a classifier that predicts the index of a particular state variable given a transition

- state variables needs to be arranged into task-independent clusters
- similarities based on state transition might not give the entire answer



# How to get rid of the expert?

## Desiderata

## Some Existing Work

**Liu & Stone 2006** Use a graph-matching algorithm to find similarities between state variables

- need for a complete and correct transition function

**Kuhlmann & Stone 2007** Similar approach tailored for GGP

**Taylor et al. 2007** Learn a classifier that predicts the index of a particular state variable given a transition

- state variables needs to be arranged into task-independent clusters
- similarities based on state transition might not give the entire answer
- learning a mapping between states versus state variables



# This talk in a single slide

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



# This talk in a single slide

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



sometimes choose a  
“transfer action”

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



sometimes choose a  
“transfer action”

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



sometimes choose a  
“transfer action”

at least as good as the  
best action according to  
current utility function

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



sometimes choose a  
“transfer action”

at least as good as the  
best action according to  
current utility function

## Outline of our approach

- 1 Using exploration to transfer knowledge
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



sometimes choose a  
“transfer action”

at least as good as the  
best action according to  
current utility function

## Outline of our approach

- 1 Using exploration to transfer knowledge  
predicts probability that  
transfer action is at least as  
good as that action
- 2 Which “transfer action” to choose?
- 3 Learn a transfer function



# This talk as a single algorithm



# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e \frac{\hat{Q}_e(s, a)}{\tau}}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```



## algorithm

- 1: initialize the Q-function hypothesis  $\hat{Q}_0$
- 2: initialize the transfer function  $\hat{P}_0$
- 3:  $e \leftarrow 0$
- 4: **repeat** {for each episode}
- 5:    $EX_Q \leftarrow \emptyset$
- 6:    $EX_P \leftarrow \emptyset$
- 7:   generate a starting state  $s_0$
- 8:    $i \leftarrow 0$
- 9:   **repeat** {for each step of episode}
- 10:     ...
- 11:   **until**  $s_i$  is terminal
- 12:   ...
- 13: **until** no more episodes
- 14:   ...
- 15:   ...
- 16:   ...
- 17:   ...
- 18:   ...

```

do
  for  $i = 0$  to  $i_{max}$  do
    sample  $x_q = (s_j, a_j, q_j)$ 
    add  $x_q$  to  $EX_Q$ 
    sample  $p$  as transfer action, i.e.
     $p(t, s_j)$  then
      sample  $a$ 
      add  $(s_j, a)$  to  $EX_P$ 
    else
      sample  $p$  as 'transfer'
      add  $(s_j, 'no transfer')$  to  $EX_P$ 
  end for
end do
update  $\hat{Q}_e$  using  $EX_Q$  to produce  $\hat{Q}_{e+1}$ 
update  $\hat{P}_e$  using  $EX_P$  to produce  $\hat{P}_{e+1}$ 
 $e = e + 1$ 
end until no more episodes

```



# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e}{\tau} \frac{\hat{Q}_e(s, a)}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```



## Transfer Learning with a single algorithm

- 9: **repeat** {for each step of episode}
- 10: **if**  $\text{rand}() < \epsilon_t$  **then** {select transfer action}
- 11:      $a_i = \pi_s(\text{argmax}_t p(t, s))$
- 12:      $t_i = \text{argmax}_t p(t, s)$  {remember transfer state}
- 13: **else** {select exploration action}
- 14:      $a_i = a$  with prob.  $\frac{e^{\frac{\hat{Q}_e(s,a)}{\tau}}}{\sum_{b \neq a} e^{\frac{Q_e(s,b)}{\tau}}}$
- 15: **end if**
- 16: take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$
- 17:  $i \leftarrow i + 1$
- 18: **until**  $s_i$  is terminal

```

← r_{i-1}
j = i - 2 to 0 do
  Q_j ← Q_j + γ q_{j+1}
  example x_q = (s_j, a_j, q_j)
  Q ← Q ∪ {x_q}
  selected as transfer action, i.e.
  if (x_t p(t, s_j)) then
    a ← a
  if (a ∈ Q_e(s_j, a)) then
    example
    'transfer')
  else
    example
    'no transfer')
  Q ← Q ∪ {x_p}
  using Exp_Q to produce Q_{e+1}
  using Exp_P to produce P_{e+1}
  e episodes
  
```



# Transfer knowledge during exploration

## The policy during exploration

- exploration policy  $\pi_e(s) : \mathcal{S} \rightarrow \mathcal{A}$ 
  - guarantees that each state-action combination has a non-zero probability of being visited
- alter exploration policy
  - motivated by previous transfer learning approaches and guidance
  - currently very simple
    - $\epsilon$ -based transfer knowledge exploration
    - with a probability of  $\epsilon$  choose a “transfer action”



## Transfer actions?

The transfer function  $p(s, t): \mathcal{S}_s \times \mathcal{S}_t \rightarrow [0, 1]$

- for each state  $s$ 
  - probability that the best action for  $s$  (*according to the source policy*)
  - is at least as good for  $t$
  - as the best action according to the current  $Q$ -function
- transfer policy  $\pi_t(t) = \pi_s(\underset{s}{\operatorname{argmax}} p(s, t))$

What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task
- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action



## Transfer actions?

The transfer function  $p(s, t): \mathcal{S}_s \times \mathcal{S}_t \rightarrow [0, 1]$

- for each state  $s$ 
  - probability that the best action for  $s$  (*according to the source policy*)
  - is at least as good for  $t$
  - as the best action according to the current  $Q$ -function
- transfer policy  $\pi_t(t) = \pi_s(\underset{s}{\operatorname{argmax}} p(s, t))$

## What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task
- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action



## Transfer actions?

The transfer function  $p(s, t): \mathcal{S}_s \times \mathcal{S}_t \rightarrow [0, 1]$

- for each state  $s$ 
  - probability that the best action for  $s$  (*according to the source policy*)
  - is at least as good for  $t$
  - as the best action according to the current  $Q$ -function
- transfer policy  $\pi_t(t) = \pi_s(\underset{s}{\operatorname{argmax}} p(s, t))$

## What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task
- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action



# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e \frac{\hat{Q}_e(s, a)}{\tau}}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```





# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e \frac{\hat{Q}_e(s, a)}{\tau}}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```



# Learning the transfer function

## Learning the transfer function

- need to learn  $p(s, t) : \mathcal{S}_s \times \mathcal{S}_t \rightarrow [0, 1]$
- at the end of every learning episode:
  - for every executed transfer action  $a_t$
  - compare 2 utility-values
    - MC based  $Q_{MC}(s, a_t)$
    - learned generalized  $\hat{Q}(s, a)$
  - learning examples
    - “transfer” if  $Q_{MC}(s, a_t) \geq \max_a \hat{Q}(s, a)$
    - and “no transfer” otherwise



# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e \frac{\hat{Q}_e(s, a)}{\tau}}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```



## This transfer

```

1: initialize the Q-function
2: initialize the transfer function
3:  $e \leftarrow 0$ 
4: repeat {for episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Ex_P \leftarrow \emptyset$ 
7:   generate example  $x_q = (s_j, a_j, q_j)$ 
8:    $i \leftarrow 0$ 
9:   repeat {for transfer}
10:    if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
11:      generate example  $x_p = (t_j, s_j, \text{'transfer'})$ 
12:       $i \leftarrow i + 1$ 
13:    else
14:      generate example  $x_p = (t_j, s_j, \text{'no transfer'})$ 
15:    end if
16:    take action  $a_j$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_j = s_{end}$ 

```

```

19:  $q_{i-1} \leftarrow r_{i-1}$ 
20: for  $j = i - 2$  to 0 do
21:    $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:   generate example  $x_q = (s_j, a_j, q_j)$ 
23:    $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:   if  $a_j$  was selected as transfer action,
       i.e.
        $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:     if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:       generate example  $x_p =$ 
            $(t_j, s_j, \text{'transfer'})$ 
27:     else
28:       generate example  $x_p =$ 
            $(t_j, s_j, \text{'no transfer'})$ 
29:     end if
30:    $Ex_P \leftarrow Ex_P \cup \{x_p\}$ 
31: end if

```

# This talk as a single algorithm

```

1: initialize the Q-function hypothesis  $\hat{Q}_0$ 
2: initialize the transfer function  $\hat{P}_0$ 
3:  $e \leftarrow 0$ 
4: repeat {for each episode}
5:    $Ex_Q \leftarrow \emptyset$ 
6:    $Exp \leftarrow \emptyset$ 
7:   generate a starting state  $s_0$ 
8:    $i \leftarrow 0$ 
9:   repeat {for each step of episode}
10:    if  $rand() < \epsilon_t$  then {select transfer action}
11:       $a_i = \pi_s(\operatorname{argmax}_t p(t, s))$ 
12:       $t_i = \operatorname{argmax}_t p(t, s)$  {remember transfer
state}
13:    else {select exploration action}
14:       $a_i = a$  with prob.  $\frac{e \frac{\hat{Q}_e(s, a)}{\tau}}{\sum_{b \neq a} e \frac{\hat{Q}_e(s, b)}{\tau}}$ 
15:    end if
16:    take action  $a_i$ , observe  $r_i$  and  $s_{i+1}$ 
17:     $i \leftarrow i + 1$ 
18:  until  $s_i$  is terminal

19:   $q_{i-1} \leftarrow r_{i-1}$ 
20:  for  $j = i - 2$  to 0 do
21:     $q_j \leftarrow r_j + \gamma q_{j+1}$ 
22:    generate example  $x_q = (s_j, a_j, q_j)$ 
23:     $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$ 
24:    if  $a$  was selected as transfer action, i.e.
 $a_j = \pi_s(\operatorname{argmax}_t p(t, s_j))$  then
25:      if  $q_j \geq \max_a \hat{Q}_e(s_j, a)$  then
26:        generate example
 $x_p = (t_j, s_j, \text{'transfer'})$ 
27:      else
28:        generate example
 $x_p = (t_j, s_j, \text{'no transfer'})$ 
29:      end if
30:       $Exp \leftarrow Exp \cup \{x_p\}$ 
31:    end if
32:  end for
33:  Update  $\hat{Q}_e$  using  $Ex_Q$  to produce  $\hat{Q}_{e+1}$ 
34:  Update  $\hat{P}_e$  using  $Exp$  to produce  $\hat{P}_{e+1}$ 
35:   $e \leftarrow e + 1$ 
36: until no more episodes

```



# Preliminary Experiment

## Preliminary Experiment

A prototype of the suggested approach

- just a proof of principle
  - *SARSA*-algorithm
  - *TG* to learn utility function
- no continuous learning of the transfer function
  - learned a transfer function once with *TILDE*
  - based on 100 episodes in the target task and actions transferred from random states in the source task



# Environments

## Source task

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(1,2)
(3,1)	(3,2)	(3,3)	(1,3)
(4,1)	(4,2)	(4,3)	<b>G</b>

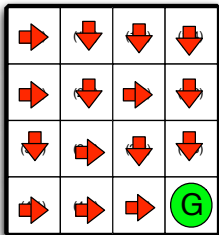
## Target task

- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal



# Environments

## Source task



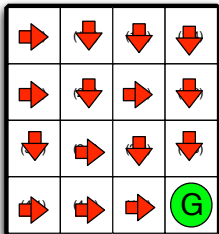
## Target task

- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal



# Environments

## Source task

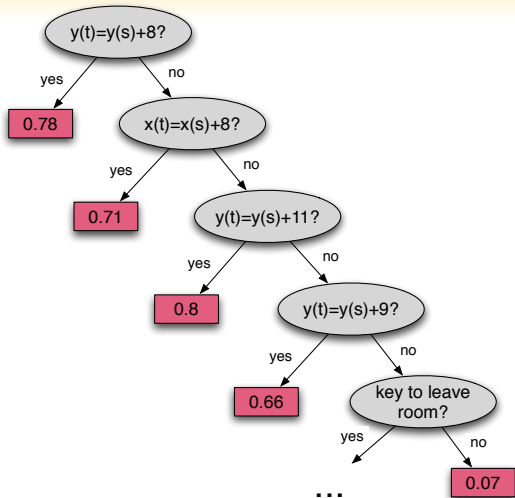


## Target task

(1,1)	(1,2)	(1,3)	(1,4)	(5,5)	(5,6)	(5,7)	(5,8)	(9,9)	(9,10)	(9,11)	(9,12)	
(2,1)	(2,2)	(2,3)	(1,2)	(6,5)	(6,6)	(6,7)	(6,8)	(10,9)	(10,10)	(10,11)	(10,12)	
(3,1)	(3,2)	(3,3)	(1,3)	(7,5)	(7,6)	(7,7)	(7,8)	(11,9)	(11,10)	(11,11)	(11,12)	
(4,1)	(4,2)	(4,3)	(4,4)	(8,5)	(8,6)	(8,7)	(8,8)	(8,8)	(12,9)	(12,10)	(12,11)	(12,12)

- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal

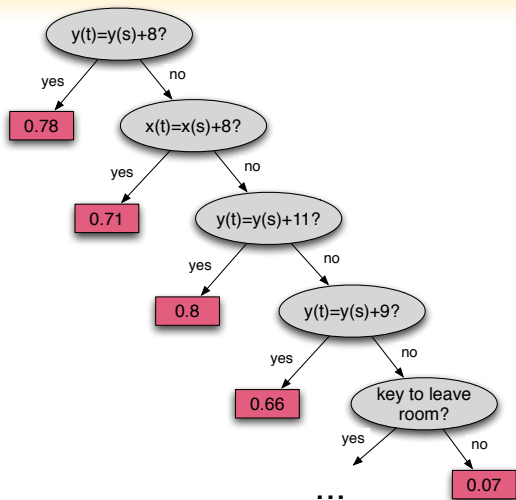
# The learned transfer function



## Able to learn both

- the shift in coordinates
- transfer only valuable if the agent has the key to leave current room

# The learned transfer function



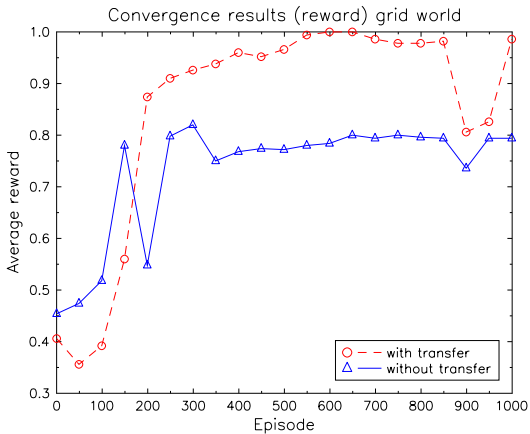
## Able to learn both

- the shift in coordinates
- transfer only valuable if the agent has the key to leave current room



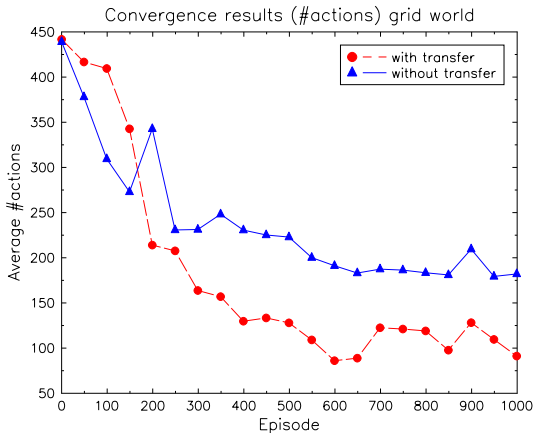
# Convergence results

## Average reward



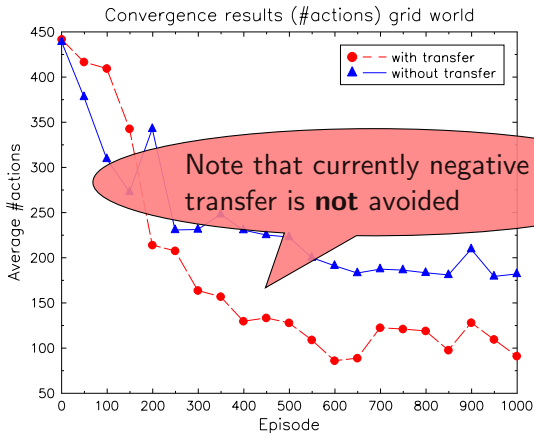
# Convergence results

## Average number of actions



# Convergence results

## Average number of actions



# Conclusions and Future Work

## Conclusions

- proposed a first step towards automatically learning a mapping function
- transfer through guided exploration
- proof of principle with promising results

## Future Work

- evaluate in more detail
- incorporate quality of possible transfer in the policy
  - avoid negative transfer
- incorporate incremental learning of the transfer function
- incorporating action descriptions into the transfer function



# Conclusions and Future Work

## Conclusions

- proposed a first step towards automatically learning a mapping function
- transfer through guided exploration
- proof of principle with promising results

## Future Work

- evaluate in more detail
- incorporate quality of possible transfer in the policy
  - avoid negative transfer
- incorporate incremental learning of the transfer function
- incorporating action descriptions into the transfer function



# Questions?

Let me start with a few....

## Questions!

- is it really a good idea to base transfer on  $Q$ -values?
- avoiding negative transfer: does  $p(s, t)$  give us enough information?
  - how low is too low?
  - measure relatively?
- substitute batch learning by a continuous incremental approach
  - what is the effect of not learning a transfer function based on random states?
  - how to deal with concept drift?
    - how quickly do you want to forget earlier examples?
- how to incorporate action descriptions into the transfer function?



# Questions?

Let me start with a few....

## Questions!

- is it really a good idea to base transfer on  $Q$ -values?
- avoiding negative transfer: does  $p(s, t)$  give us enough information?
  - how low is too low?
  - measure relatively?
- substitute batch learning by a continuous incremental approach
  - what is the effect of not learning a transfer function based on random states?
  - how to deal with concept drift?
    - how quickly do you want to forget earlier examples?
- how to incorporate action descriptions into the transfer function?

