L2-Matlab

L2-Matlab

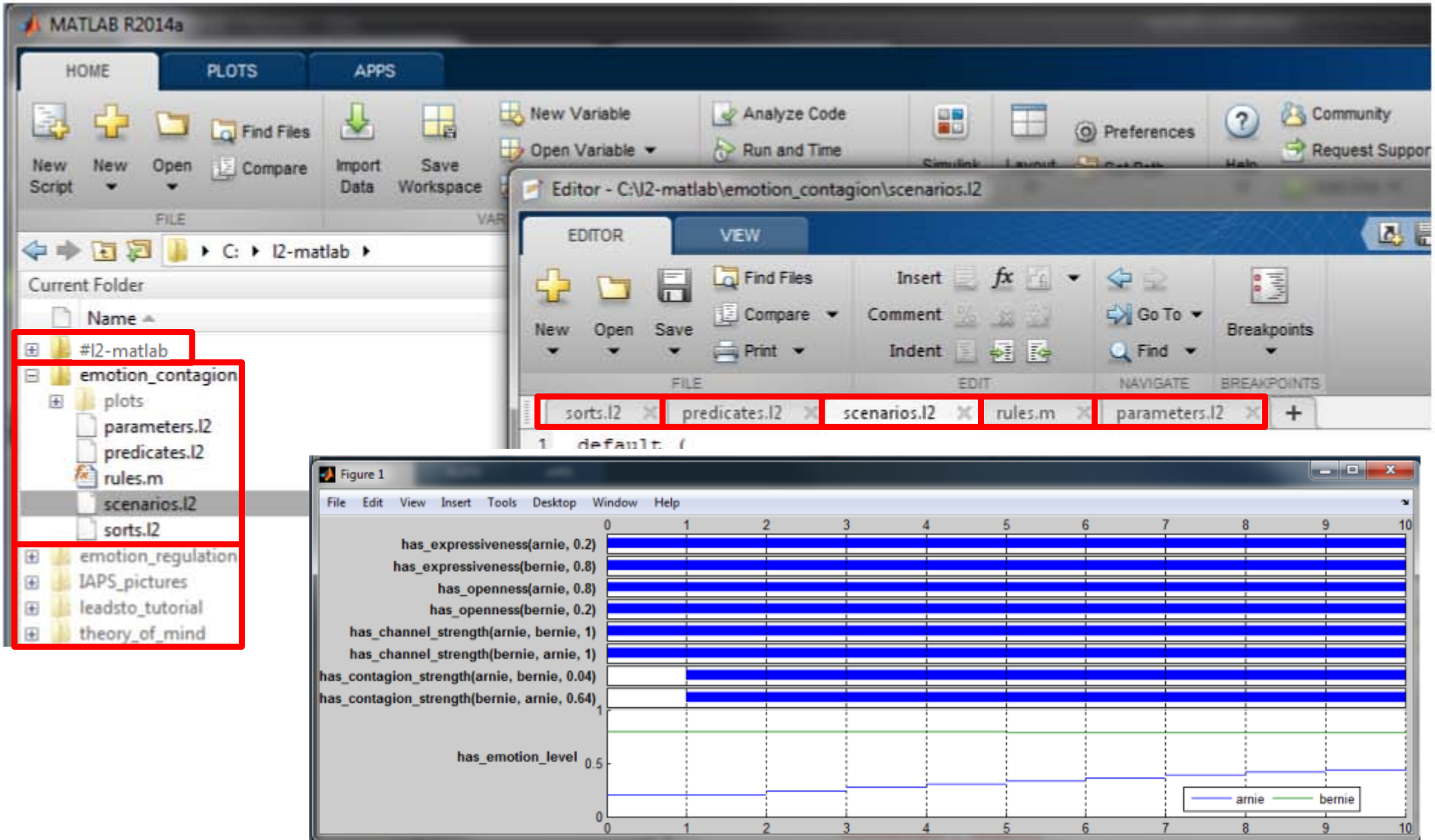
- Introduction
- First time use
- Overview
- Models in Matlab
- l2-matlab functions
- Writing rules
- Example: IAPS Pictures
- Conclusion

Introduction

- Collection of **Matlab classes and functions**
- **Sorts & predicates** define static model
- Model dynamics are written as **rules**
- **Scenario** defines simulation instantiation
- **Parameters** can be used for convenience

- **L2-matlab** simulates and plots these model definitions

Introduction

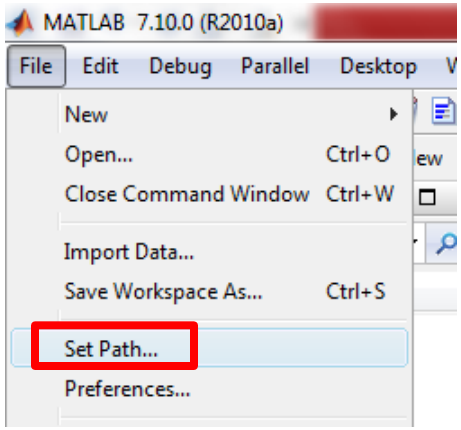


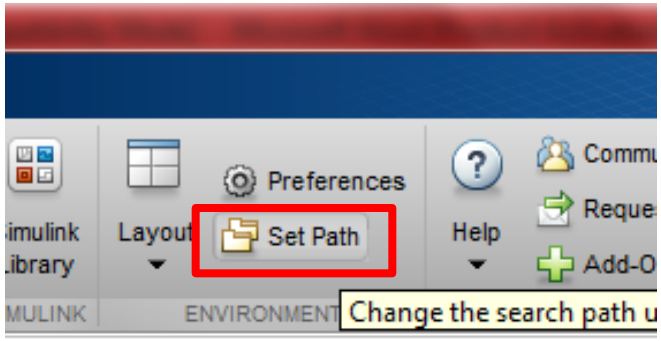
L2-Matlab

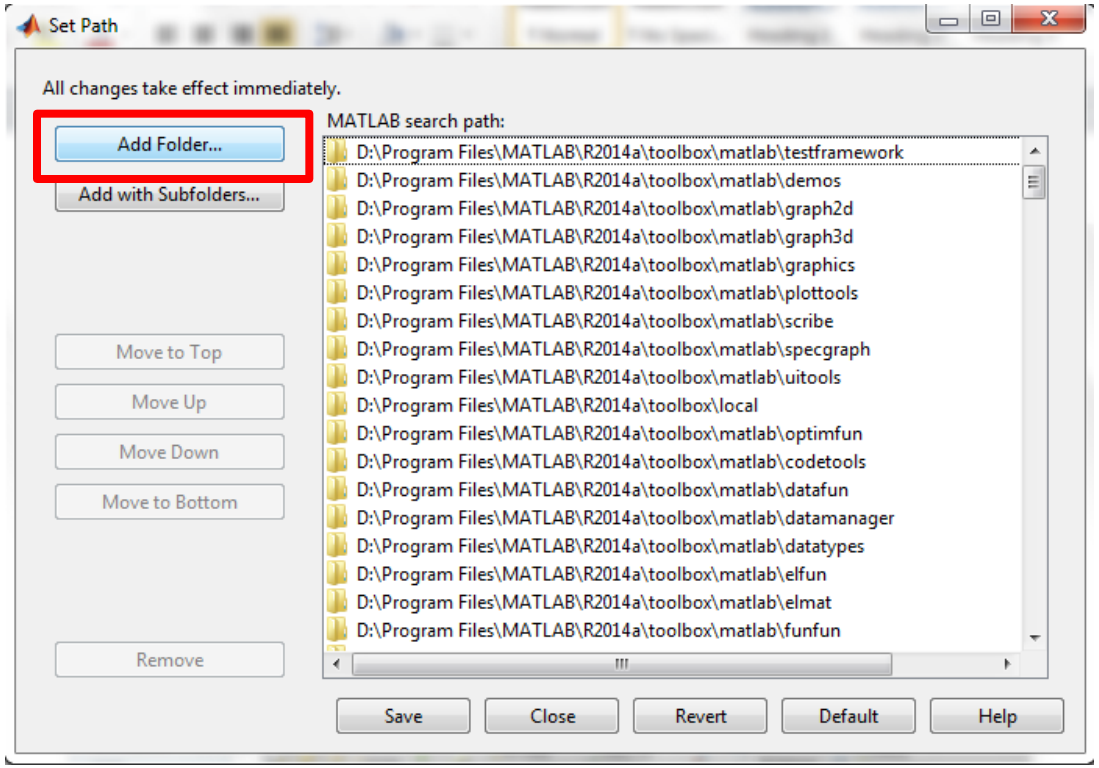
- Introduction
- **First time use**
  - Installation
  - Folder structure
- Overview
- Models in Matlab
- l2-matlab functions
- Writing rules
- Example: IAPS Pictures
- Conclusion

Installation

Download, unzip & add **#l2-matlab** to **Path**







Folder structure

- Each model resides in its own folder
- Only one model per folder!

folder structure

- 00_intro_slides
  - sorts.l2
  - predicates.l2
  - scenarios.l2
  - parameters.l2
  - rules.m
- 01_leadsto_tutorial
- 02_emotion_regulation
- …

L2-Matlab

- Introduction
- First time use
- **Overview**
  - sorts.l2
  - predicates.l2
  - scenarios.l2
  - parameters.l2
  - rules.m
- Models in Matlab
- l2-matlab functions
- Writing rules
- Example: IAPS Pictures
- Conclusion

sorts.l2

- Sorts define a set of instances
- REAL & BOOLEAN are built-in

sorts.l2

```
AGENT;      {arnie, bernie, charlie}
```

predicates.l2

- Predicates are relations between sorts

predicates.l2

```
couple;              {AGENT, AGENT}
married;             BOOLEAN
nr_children;         REAL
child;               AGENT
```

scenarios.l2

- A scenario defines the initial values for a simulation

scenarios.l2

```
couple{arnie, bernie};    [1]
married{false};           [1]
nr_children{0};           [1]
```

scenarios.l2

- A scenario defines the initial values for a simulation
- Multiple scenarios can be defined

scenarios.l2

```
default(
   couple{arnie, bernie};        [1]
   married{false};               [1]
   nr_children{0};               [1]
)
alternative(
   couple{arnie, bernie};        [1]
   married{false};               [1]
   nr_children{1};               [3]
   child{arnie};                 [3]
)
```

parameters.l2

- Parameters are defined for the entire simulation

parameters.l2

```
adult_age;          18
```

parameters.l2

- Parameters are defined for the entire simulation
- Multiple sets of parameters can be defined

parameters.l2

```
default(
   adult_age;      18
)

US(
  adult_age;       19
)
```

rules.m

- Define the dynamics of the model
- Each function executed once for each time step

rules.m

```
function [ fncs ] = rules()
    % DO NOT EDIT
    fncs = l2.getRules();
    for i=1:length(fncs)
        fncs{i} = str2func(fncs{i});
    end
end

%ADD RULES BELOW
function result = ddr1( trace, parameters, t )
    …
end
…
```

Any Matlab code allowed to derive your result!

Parameter values

Simulation results up to and including t

Current time point

L2-Matlab

- Introduction
- First time use
- Overview
- **Models in Matlab**
  - Model
  - Predicates
  - Trace
- l2-matlab functions
- Writing rules
- Example: IAPS Pictures
- Conclusion

Model

- The model is an instance of the class l2
- The folder name is used to specify the exact model

matlab

```
>> model = l2('00_family')

model =
  l2 with properties:

         model: 'D:\Documents\Dropbox\#VU\l2-matlab\00_family'
         trace: [0x0 struct]
         sorts: [1x1 struct]
    predicates: [1x1 struct]
    parameters: [1x1 struct]
     scenarios: [1x1 struct]
         files: [1x1 struct]
```

Model

- Sorts and predicates are read from the .l2 files

matlab

```
>> model.sorts

ans =
    AGENT: {'arnie'  'bernie'  'charlie'}

>> model.predicates

ans =
          couple: {'AGENT'  'AGENT'}
         married: 'BOOLEAN'
    nr_children: 'REAL'
           child: 'AGENT'
```

Model

- Similarly, the various scenarios and parameter sets can be accessed

matlab

```
>> model.scenarios

ans =
        default: [1x1 struct]
    alternative: [1x1 struct]

>> model.parameters

ans =
    default: [1x1 struct]
         US: [1x1 struct]
```

Model

- Also, the file names used can be shown *(and altered if required, see manual)*

matlab

```
>> model.files

ans =
          rules: 'rules.m'
          sorts: 'sorts.l2'
     predicates: 'predicates.l2'
      scenarios: 'scenarios.l2'
     parameters: 'parameters.l2‘
```

Model

- A model can be simulated and plotted, results are stored in the trace.

matlab

```
>> model.simulate(5);
>> model.plot();

>> model.trace

ans =
1x5 struct array with fields:

    couple
    married
    nr_children
    child
```

Predicate

- A predicate is an instance of the class predicate
- It has a *name* and arguments *arg*

matlab

```
>> p = predicate('couple{arnie,bernie}')

p =

  predicate with properties:

    name: 'couple'
     arg: {'arnie'  'bernie'}

>> p = predicate('couple', {'arnie','bernie'});
```

Predicate

- Retrieve its name via the respective field
- Access to its values via the cell array in the field arg

matlab

```
>> p.name

ans =
couple

>> p.arg{2}

ans =
bernie
```

Predicate

- Predicate with single REAL or BOOLEAN value can be used directly for simple arithmetic and logic

matlab

```
>> p = predicate('nr_children{1}');
>> p > 0

ans =
     1

>> p + 1

ans =
     2
```

Trace

- Structure array with fields for each predicate type
- Dimensions represent time points

matlab

```
>> model.trace

ans =
1x5 struct array with fields:

    couple
    married
    nr_children
    child
```

Trace

- For each time point, each field contains an array of predicates that hold for that time point

matlab

```
>> model.trace(1)

ans =
          couple: [1x1 predicate]
         married: [1x1 predicate]
    nr_children: [1x1 predicate]
           child: []
```

Trace

- For each time point, each field contains an array of predicates that hold for that time point

matlab

```
>> model.trace(1).couple(1)

ans =
  predicate with properties:

    name: 'couple'
     arg: {'arnie'  'bernie'}

>> model.trace(1).couple(1).arg{1}

ans =
arnie
```

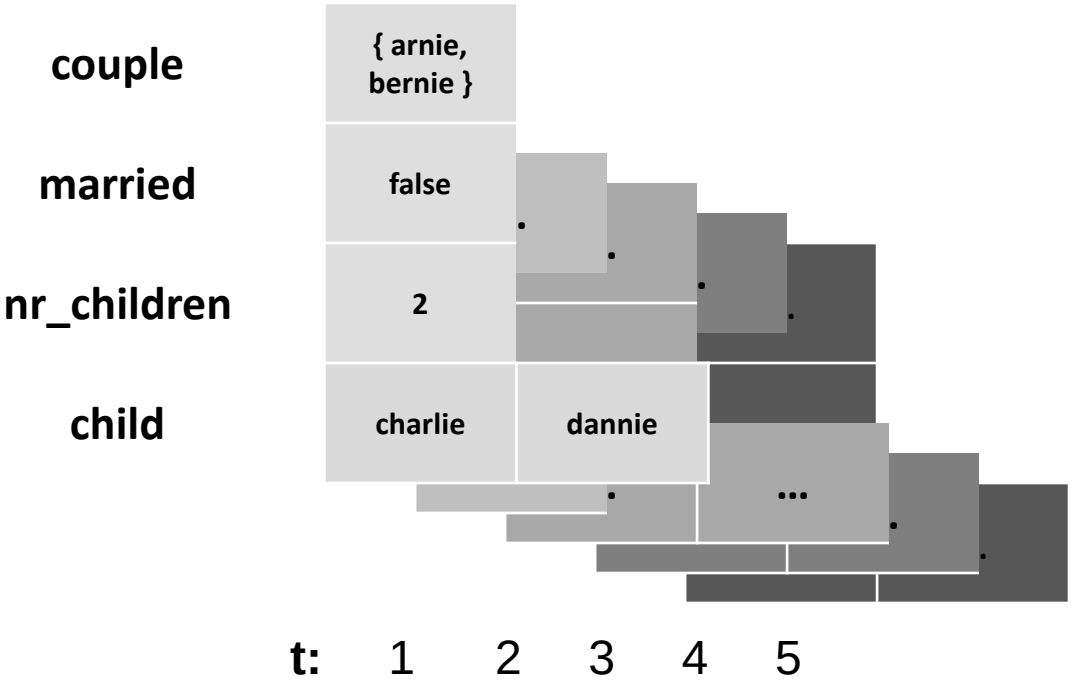
Trace

visualised

couple	**{ arnie, bernie }**
married	**false**
nr_children	**0**
child	

Trace

visualised



L2-Matlab

- Introduction
- First time use
- Overview
- Models in Matlab

- **l2-matlab functions**
  - l2.exists
  - l2.forall
  - str2pred, pred2str
  - ispredicate, predcmp
- Writing rules
- Example: IAPS Pictures
- Conclusion

l2.exists

- l2.exists to check if a particular value exists
- Pattern uses ~ or NaN if value may be anything

matlab

Pattern of target predicate

```
[tf, results] = l2.exists(trace, t, 'couple{arnie, bernie}')
```

l2.exists

- l2.exists to check if a particular value exists
- Pattern uses ~ or NaN if value may be anything

matlab

Pattern of target predicate

```
[tf, results] = l2.exists(trace, t, 'couple{arnie, bernie}')

[tf, results] = l2.exists(trace, t, 'couple{arnie, ~}')

[tf, results] = l2.exists(trace, t, 'couple', {'arnie', NaN})

p = predicate('couple{arnie, ~}');
[tf, results] = l2.exists(trace, t, p)
```

l2.exists

- Commonly used for if statements

matlab

```
if l2.exists(trace, t, 'couple{arnie, bernie}')
        ...
end
```

l2.getall

- l2.getall to get all values for a particular predicate
- Commonly used in for loops

matlab

```
results = l2.getall(trace, t, 'couple{arnie, ~}')
```

l2.getall

- l2.getall to get all values for a particular predicate
- Commonly used in for loops

matlab

```
results = l2.getall(trace, t, 'couple{arnie, ~}')


for p = l2.getall(trace, t, 'couple{arnie, ~}')
        ...
end
```

str2pred, pred2str

- str2pred to convert string in predicate
- pred2str to convert predicate to string

matlab

```
>> p = str2pred('couple{arnie, ~}')

p =
  predicate with properties:

    name: 'couple'
     arg: {'arnie'  [NaN]}
```

str2pred, pred2str

- str2pred to convert string in predicate
- pred2str to convert predicate to string

matlab

```
>> p = str2pred('couple{arnie, ~}')

p =
  predicate with properties:

    name: 'couple'
     arg: {'arnie'  [NaN]}

>> pred2str(p)

ans =
couple{arnie, ~}
```

ispredicate, predcmp

- ispredicate to check if a variable is of type predicate
- predcmp to check if two predicates are similar

matlab

```
>> p = str2pred('couple{arnie, ~}');
>> ispredicate(p)

ans =
     1
```

ispredicate, predcmp

- ispredicate to check if a variable is of type predicate
- predcmp to check if two predicates are similar

matlab

```
>> p = str2pred('couple{arnie, ~}');
>> ispredicate(p)

ans =
     1

>> predcmp(p, 'couple{arnie, bernie}')

ans =
     1
```

L2-Matlab

- Introduction
- First time use
- Overview
- Models in Matlab

- l2-matlab functions
- **Writing rules**
  - l2.exists
  - l2.forall
  - str2pred, pred2str
  - ispredicate, predcmp
- Example: IAPS Pictures
- Conclusion

Writing rules

- Trace, parameters and current time point given
- Results are cell arrays with time point and predicate

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, parameters, t )




end
```

Parameter values

Simulation results up to and including t

Current time point

Writing rules

- Trace, parameters and current time point given
- Results are cell arrays with time point and predicate

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, parameters, t )

    result = {t+1, 'couple{arnie, bernie}'};
end
```

Parameter values

Simulation results up to and including t

Current time point

Resulting predicate

Time point of result

Writing rules

- Trace, parameters and current time point given
- Results are cell arrays with time point and predicate

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, parameters, t )

    result = {t+1, 'couple', {'arnie', 'bernie'}};
end
```

Parameter values

Simulation results up to and including t

Current time point

Resulting predicate

Time point of result

Writing rules

- Trace, parameters and current time point given
- Results are cell arrays with time point and predicate

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, parameters, t )


    p = predicate('couple{arnie, bernie}');
    result = {t+1, p};
end
```

Parameter values

Simulation results up to and including t

Current time point

Resulting predicate

Time point of result

Writing rules

- Multiple results can be returned in a cell array
- Often used by adding new results to an existing array

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, parameters, t )
    result = {};

    for p = l2.forall(trace, t, 'couple{arnie, ~}')
       ...
       result = {result{:}, {t+1, p}};
    end


end
```

L2-Matlab

- Introduction
- First time use
- Overview
- Models in Matlab
- l2-matlab functions
- Writing rules
- **Example: IAPS Pictures**
- Conclusion

Example: IAPS pictures

description

'The International Affective Picture System (IAPS) is [..] a set of normative emotional stimuli for experimental investigations of emotion and attention.' (Lang et al., 1999) Each of these pictures has a rating for both valence and arousal. Consider an agent viewing a number of these pictures. The valance and arousal experienced by that agent will adept to those of the pictures viewed. The speed of this process depends on the speed parameters w_1 and w_2 respectively.

DDR1 (Domain Dynamic Relation 1)

at any point in time, for each agent X
if agent X has a valence V and arousal A
and an agent X observes a picture P from the IAPS picture set
and picture P has a valance rating V1 and arousal rating A1
then agent X will get a valence of $V+w_1(V1-V)$ and arousal of $A+w_2(A1-A)$

Example: IAPS pictures

sorts.l2

```
AGENT;        {arnie, bernie, charlie}
PICTURE;      {p1, p2, p3}
```

Example: IAPS pictures

predicates.l2

```
% picture(PICTURE, VALENCE, AROUSAL}
picture;    {PICTURE, REAL, REAL}

% emotion(AGENT, VALENCE, AROUSAL)
emotion;    {AGENT, REAL, REAL}

observes;   {AGENT, PICTURE}
```

Example: IAPS pictures

scenarios.l2

```
% IAPS picture valence/arousal
picture{p1, 1, 9};       [1:Inf]
picture{p2, 9, 9};       [1:Inf]

% Starting emotion of the agents
emotion{arnie, 4.5, 4.5};     1
emotion{bernie, 4.5, 4.5};    1

% Observing of pictures
observes{arnie, p1};     [1:25]
observes{bernie, p1};    [1:25]
observes{arnie, p2};     [26:50]
```

Example: IAPS pictures

parameters.l2

```
w1;             0.1
w2;             0.1
```

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion









  end
end
```

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion
    x = emotion.arg{1}; %agent name
    v = emotion.arg{2}; %valence
    a = emotion.arg{3}; %arousal









  end
end
```

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion
    x = emotion.arg{1}; %agent name
    v = emotion.arg{2}; %valence
    a = emotion.arg{3}; %arousal
    % go through each picture that agent observes
    for observes = l2.getall(trace, t, 'observes', {x, NaN})
      p = observes.arg{2}; %picture name









    end


  end
end
```

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion
    x = emotion.arg{1}; %agent name
    v = emotion.arg{2}; %valence
    a = emotion.arg{3}; %arousal
    % go through each picture that agent observes
    for observes = l2.getall(trace, t, 'observes', {x, NaN})
      p = observes.arg{2}; %picture name
      %get the valence and arousal of that picture
      [~, pic] = l2.exists(trace, t, 'picture', {p, NaN, NaN});
      v1 = pic.arg{2}; %picture valence
      a1 = pic.arg{3}; %picture arousal



    end


  end
end
```

rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion
    x = emotion.arg{1}; %agent name
    v = emotion.arg{2}; %valence
    a = emotion.arg{3}; %arousal
    % go through each picture that agent observes
    for observes = l2.getall(trace, t, 'observes', {x, NaN})
      p = observes.arg{2}; %picture name
      %get the valence and arousal of that picture
      [~, pic] = l2.exists(trace, t, 'picture', {p, NaN, NaN});
      v1 = pic.arg{2}; %picture valence
      a1 = pic.arg{3}; %picture arousal
      %adjust v and a accordingly
      v = v + params.w1 * (v1 - v);
      a = a + params.w2 * (a1 - a);
    end


  end
end
```

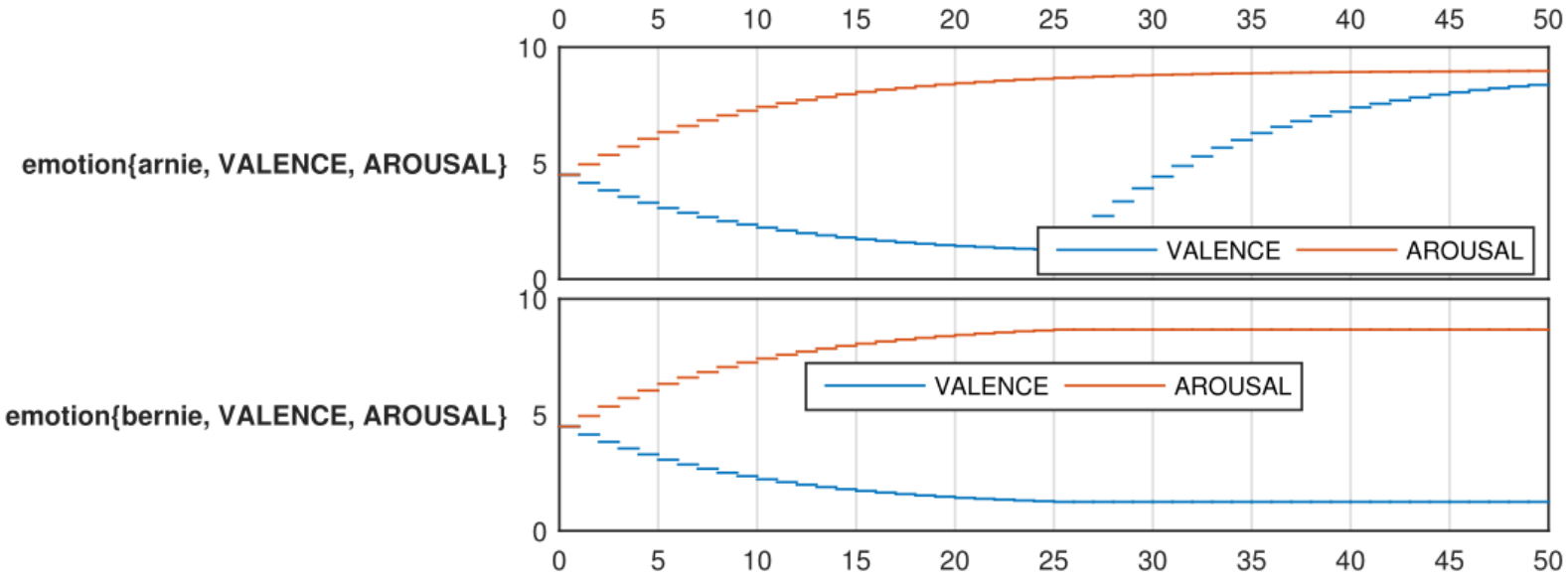
rules.m

```
%ADD RULES BELOW
function result = ddr1( trace, params, t )
  result = {};
  %go through each agent's emotion
  for emotion = trace(t).emotion
    x = emotion.arg{1}; %agent name
    v = emotion.arg{2}; %valence
    a = emotion.arg{3}; %arousal
    % go through each picture that agent observes
    for observes = l2.getall(trace, t, 'observes', {x, NaN})
      p = observes.arg{2}; %picture name
      %get the valence and arousal of that picture
      [~, pic] = l2.exists(trace, t, 'picture', {p, NaN, NaN});
      v1 = pic.arg{2}; %picture valence
      a1 = pic.arg{3}; %picture arousal
      %adjust v and a accordingly
      v = v + params.w1 * (v1 - v);
      a = a + params.w2 * (a1 - a);
    end
    %add the new emotion level to the results
    result = { result{:} {t+1, 'emotion', {x, v, a}} };
  end
end
```

Example: IAPS pictures

matlab

```
clear all
close all
model = l2('03_IAPS_pictures')
model.simulate(50);
model.plot({'emotion'});
```



L2-Matlab

- Introduction
- First time use
- Overview
- Models in Matlab
- l2-matlab functions
- Writing rules
- Example: IAPS Pictures
- **Conclusion**

Conclusion

- Introduction to using l2-matlab
- Not covered:
  - Simulating alternative scenarios / parameter sets
  - Plotting to files
  - Nested predicates
  - Custom plotting functions *(advanced)*
- Read on:
  - Examples, walkthrough, manual, Matlab tutorials

http://www.few.vu.nl/~jmn300/l2-matlab/