BDN-Format

Das BDN-Dateiformat des BrainDesigners ist ein auf XML basierendes Format zum Speichern neuronaler Netze für den BrainDesigner.

Das Wurzelelement ist `BrainDesigner`, es enthält beliebig viele `Module`-Elemente.

Die Dokumenttypdefinition (DTD) lautet wie folgt:

```
<!ELEMENT BrainDesigner (Module)*>
<!ATTLIST BrainDesigner
   version       CDATA   #REQUIRED
   libraryname   CDATA   #REQUIRED
   librarycolor  CDATA   #IMPLIED
>

<!ELEMENT Module (bytecode, IoLabelList, node, edge)*>
<!ATTLIST Module
   name          CDATA   #REQUIRED
   type          CDATA   #REQUIRED
   guid          CDATA   #REQUIRED    filename  CDATA
   #IMPLIED
   show          CDATA   #IMPLIED
>

<!ELEMENT bytecode #PCDATA>

<!ELEMENT IoLabelList (IoLabel)*>
<!ATTLIST IoLabelList
   name          CDATA   #REQUIRED
>

<!ELEMENT IoLabel EMPTY>
<!ATTLIST IoLabel
   name          CDATA   #REQUIRED
   standard      CDATA   #REQUIRED
>

<!ELEMENT node (position, connectedEdges, parameters?,
curves?)>
<!ATTLIST node
   id            CDATA   #REQUIRED
   type          CDATA   #REQUIRED
   guid          CDATA   #REQUIRED
   comment       CDATA   #IMPLIED
>

<!ELEMENT position EMPTY>
<!ATTLIST position
   x             CDATA   #REQUIRED
   y             CDATA   #REQUIRED
>

<!ELEMENT connectedEdges (connectionPoint)*>
```

```
<!ELEMENT connectionPoint (connectedEdge)*>
<!ATTLIST connectionPoint
   id            CDATA   #REQUIRED
>

<!ELEMENT connectedEdge EMPTY>
<!ATTLIST connectedEdge
   id            CDATA   #REQUIRED
   start         CDATA   #REQUIRED
>

<!ELEMENT parameters (parameter)*>

<!ELEMENT parameter EMPTY>
<!ATTLIST parameter
   name          CDATA   #REQUIRED
   value         CDATA   #REQUIRED
>

<!ELEMENT curves (curve)*>

<!ELEMENT curve EMPTY>
<!ATTLIST curve
   output        CDATA   #REQUIRED
   color         CDATA   #REQUIRED
   subcurve      CDATA   #REQUIRED
>

<!ELEMENT edge (start, end, roundness, parameters?,
curves?)>
<!ATTLIST edge
   id            CDATA   #REQUIRED
   guid          CDATA   #REQUIRED
   comment       CDATA   #IMPLIED
>

<!ELEMENT start EMPTY>
<!ATTLIST start
   nodeId        CDATA   #IMPLIED
   x             CDATA   #IMPLIED
   y             CDATA   #IMPLIED
>

<!ELEMENT end EMPTY>
<!ATTLIST end
   nodeId        CDATA   #IMPLIED
   x             CDATA   #IMPLIED
   y             CDATA   #IMPLIED
>

<!ELEMENT roundness EMPTY>
<!ATTLIST roundness
   height        CDATA   #REQUIRED
   width         CDATA   #REQUIRED
>
```

Beschreibung der Xml-Elemente

Module

Ein Module ist entweder eine durch Bytecode beschriebene Recheneinheit oder eine grafische Representation eines neuronalen Netzes, das sich aus anderen Modules zusammensetzt.

Ist der type eines Modules „Structure“, enthält es ausschließlich Elemente des Types „node“ und „edge“, die den grafischen Aufbau des Netzes repräsentieren. Ist der type „Unit“ oder „Synapse“ besteht das Xml-Element nur aus einem bytecode-Element und IoLabelList-Elementen (mit den name-Attributen „Inputs“, „Outputs“, „Parameters“ oder „Internals“). Bei System-Elementen gibt es ausschließlich ein IoLabelList-Element (name=„Parameters“).

Ein Module-Element hat vier Attribute: `name`, `type`, `guid`, `opened`.

name

Ist der angezeigte Name des Modules. Wird nur zur Identifizierung durch den Nutzer verwendet.

type

Einer der folgenden Werte: `Structure`, `Unit`, `Synapse`, `System`, `Library`. `Structure` ist eine grafische Netz-Repräsentation, `Unit` ist eine Knoten-Recheneinheit, `Synapse` eine Kanten-Recheneinheit und `System` wird für Systemelemente wie Inputs und Outputs verwendet. `Library` bedeutet, dass eine andere Library eingebunden ist. Im BrainDesigner ist die Verwendung des `Library`-Types nur in der Hauptbibliothek erlaubt.

guid

Der Globally Unique Identifier (guid) muss ein über alle BrainDesigner-Bibliotheken eindeutiger String sein.

opened

Ist das opened-Attribut angegeben und auf „True“ gesetzt, wird dieses Element als geöffnet betrachtet und beim nächsten Start des BrainDesigners geöffnet.

filename

Ist der type des Modules `Library`, wird hier der Dateiname der eingebundenen Bibliothek angegeben.

show

Ist der type des Modules `Library`, wird hier angegeben, ob die Bibliothek ein- oder ausgeklappt dargestellt wird.

Structure-Module

Im Folgenden wird ein Beispiel-Structure-Module dargestellt. Es enthält verschiedene nodes und edges, die alle vom Typ „TanhNeuron“ bzw. „StandardSynapse“ sind. Zusammen bilden sie einen SO(2)-Oszillator.

```
  <Module name="SO(2)" type="Structure" guid="99246760-026a-40c2-8601-
5b7d38f77862" opened="true">
```

Das Module hat den Namen „SO(2)“ und ist das aktuell geöffnete Module. Es hat eine eindeutige guid.

```
    <node id="0" type="Unit" guid="TanhNeuron" comment="Ein Kommentar">
      <position x="462" y="297" />
      <connectedEdges>
        <connectionPoint id="1">
          <connectedEdge id="3" start="False" />
        </connectionPoint>
        <connectionPoint id="3">
          <connectedEdge id="3" start="True" />
        </connectionPoint>
        <connectionPoint id="4">
          <connectedEdge id="4" start="False" />
        </connectionPoint>
        <connectionPoint id="5">
          <connectedEdge id="0" start="True" />
        </connectionPoint>
        <connectionPoint id="7">
          <connectedEdge id="1" start="False" />
        </connectionPoint>
      </connectedEdges>
    </node>
```

Dies ist das erste node-Element. Es ist eine Unit mit der ID 0 und eine Instanz des Moduls „TanhNeuron“. Zusätzlich ist ein optionaler Kommentar angegeben. Mit dem Kindelement „position“ ist die grafische Position des Elements innerhalb des SO(2)-Modules angegeben. Über das Element connectedEdge und dessen Kindelemente ist angegeben, welche edge-Elemente des SO(2)-Modules an dieses node-Element angeschlossen sind. Die id des connectionPoint-Elements zeigt dabei an, wo eine Synapse angeschlossen ist. Bei Units ohne benannte Inputs und Outputs, die als Kreis dargestellt werden (wie in diesem Beispiel), gibt es acht connection points, die unten beginnend im 45°-Abstand im Uhrzeigersinn um die Unit gehen. Hat eine Unit benannte In- und Outputs oder handelt es sich um eine andere Struktur (mit benannten In- und Outputs), beginnt die Nummerierung der connection points mit den Inputs und setzt sich mit den Outputs fort.

In diesem Beispiel ist an den ersten beiden connectionPoints erkennbar, dass die edge mit der id „3“ eine rekurrente Synapse ist, die von connection point 3 zu connection point 1 führt.

```
    <node id="1" type="Unit" guid="TanhNeuron">
      <position x="778" y="295" />
      <connectedEdges>
        <connectionPoint id="1">
          <connectedEdge id="1" start="True" />
        </connectionPoint>
        <connectionPoint id="3">
          <connectedEdge id="0" start="False" />
        </connectionPoint>
        <connectionPoint id="4">
          <connectedEdge id="5" start="True" />
        </connectionPoint>
        <connectionPoint id="5">
          <connectedEdge id="2" start="True" />
        </connectionPoint>
        <connectionPoint id="7">
          <connectedEdge id="2" start="False" />
        </connectionPoint>
      </connectedEdges>
    </node>
    <node id="2" type="Bias" guid="Bias">
      <position x="305" y="161" />
      <connectedEdges>
```

```
      <connectionPoint id="1">
        <connectedEdge id="4" start="True" />
      </connectionPoint>
    </connectedEdges>
  </node>
```

Die node mit der id 2 ist ein Bias-Element. Es hat drei potentielle connection points (0 bis 2). Logischerweise sollte das start-Attribut aller connected edges bei einem Bias „true“ sein.

```
  <node id="3" type="Output" guid="Output">
    <position x="919" y="149" />
    <connectedEdges>
      <connectionPoint id="1">
        <connectedEdge id="5" start="False" />
      </connectionPoint>
    </connectedEdges>
    <parameters>
      <parameter name="Position" value="M29" />
      <parameter name="Output Type" value="M-Series" />
    </parameters>
  </node>
```

Dies ist ein Output-Knoten. Bei Outputs sind natürlich sämtliche start-Attribute von connected edges „false“. Zusätzlich werden hier Parameter angegeben. Der „Output Type“ ist hier „M-Series“. Weitere Parameter hängen vom Output Type ab. Bei „M-Series“ muss zusätzlich eine „Position“ angegeben werden, hier M29 für Motor 29.

```
  <edge id="0" guid="StandardSynapse" comment="Das ist die Ringkopplung">
    <start nodeId="0" />
    <end nodeId="1" />
    <roundness height="-29" width="-1" />
    <parameters>
      <parameter name="w" value="0.2" />
    </parameters>
  </edge>
```

Dies ist die erste Synapse. Wieder kann ein Kommentar angegeben werden. Start und end geben an, von welchem Knoten zu welchem Knoten diese Synapse führt. Das Xml-Element roundness gibt die Krümmung des Pfeils an. Dieser Synapsen-Typ hat einen Parameter, dessen Wert hier auf 0.2 gesetzt ist.

```
  <edge id="1" guid="StandardSynapse">
    <start nodeId="1" />
    <end nodeId="0" />
    <roundness height="-18" width="0" />
    <parameters>
      <parameter name="w" value="-0.2" />
    </parameters>
  </edge>
  <edge id="2" guid="StandardSynapse">
    <start nodeId="1" />
    <end nodeId="1" />
    <roundness height="-38" width="49" />
    <parameters>
      <parameter name="w" value="1.1" />
    </parameters>
  </edge>
  <edge id="3" guid="StandardSynapse">
    <start nodeId="0" />
    <end nodeId="0" />
    <roundness height="37" width="67" />
    <parameters>
      <parameter name="w" value="1.1" />
```

```
      </parameters>
    </edge>
    <edge id="4" guid="StandardSynapse">
      <start nodeId="2" />
      <end nodeId="0" />
      <roundness height="-39" width="3" />
      <parameters>
        <parameter name="w" value="0.001" />
      </parameters>
    </edge>
    <edge id="5" guid="StandardSynapse">
      <start nodeId="1" />
      <end nodeId="3" />
      <roundness height="-42" width="-4" />
      <parameters>
        <parameter name="w" value="0.6" />
      </parameters>
    </edge>
    <edge id="6" guid="StandardSynapse">
      <start x="523" y="453" />
      <end x="623" y="453" />
      <roundness height="0" width="0" />
    </edge>
  </Module>
```

Die letzte Synapse ist eine nicht verbundene. Statt nodeIds werden in den start- und end-Elementen x- und y-Positionen angegeben.

Unit-/Synapse-Module

Im Folgenden ist jeweils ein Unit-Module und ein Synapse-Module angegeben:

```
  <Module name="TanhNeuron" type="Unit" guid="TanhNeuron">
    <bytecode>300:
  tanh  V0, Input
  write Output, V0
</bytecode>
  </Module>
  <Module name="StandardSynapse" type="Synapse" guid="StandardSynapse">
    <bytecode>200:
  mul   V0, Input, w
  write Output, V0
</bytecode>
    <IoLabelList name="Parameters">
      <IoLabel name="w" standard="1" />
    </IoLabelList>
  </Module>
```

Das Unit-Module besitzt ausschließlich Bytecode, das Synapse-Module zusätzlich einen im Bytecode verwendeten Parameter („w“), dem zusätzlich ein Standardwert (hier „1“) zugewiesen wird.