ANN Circus Rules

Ziggy Attala

March 2024

1 Z Specification of ANNControllers

$\textbf{section}\ \mathit{spec}\ \textbf{parents}\ \mathit{standard\_toolkit}$

We just assume we have a context, then we can run the allEvents metalanguage function on this context.

$[\mathit{Context}, \mathit{CSExp}, \mathit{Channel}, \mathit{CircusProgram}]$

$\mathit{ActivationFunction} ::= \mathit{RELU} \mid \mathit{LINEAR} \mid \mathit{NOTSPECIFIED}$

Type synomym for ANNs.

$\mathit{Value} == \mathbb{A}$

Sequence of values, if 1d, then it is layerstructure, 2d or 3d, biases or layer structure.

$\mathit{SeqExp} ::= \mathit{null\_seq} \mid \mathit{list}\langle\langle \mathrm{seq}\,\mathbb{N} \rangle\rangle \mid \mathit{matrix}\langle\langle \mathrm{seq}\,\mathrm{seq}\,\mathit{Value} \rangle\rangle \mid \mathit{tensor}\langle\langle \mathrm{seq}\,\mathrm{seq}\,\mathrm{seq}\,\mathit{Value} \rangle\rangle$

$$
\begin{array}{|l}
\mathit{relu} : \mathit{Value} \rightarrow \mathit{Value} \\
\hline
\forall x : \mathit{Value} \bullet \\
\quad (x < 0 \Rightarrow (x, 0) \in \mathit{relu}) \land \\
\quad (x \geq 0 \Rightarrow (x, x) \in \mathit{relu})
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{ANNParameters} \\
\mathit{layerstructure} : \mathit{SeqExp} \\
\mathit{weights} : \mathit{SeqExp} \\
\mathit{biases} : \mathit{SeqExp} \\
\mathit{activationfunction} : \mathit{ActivationFunction} \\
\mathit{inputContext} : \mathit{Context} \\
\mathit{outputContext} : \mathit{Context} \\
\hline
\end{array}
$$

To match the meta-model better.

$$\begin{array}{|l}
\multicolumn{1}{l}{_ \; ANN \; ____________________________________} \\
annparameters : ANNParameters \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{_ \; ANNController \; ____________________________} \\
ANN \\
\hline
\end{array}$$

2 Semantic Rules

NOTES:

- We use *Value* as a meta-language type, we define it in our Z Specification, but we also use it in our Circus program, they are different, in Circus, it is specialised to the real number type. We keep this, and not just $\mathbb{R}$, to support BNNs, or other types of ANNs, or other precisions of ANNs.
- $\{\!|$ and $|\!\}$ brackets, used in rule 16, for example, in the metalanguage, for us, it is in the target language, but how do we express, the set in between, it is all the elements, in the set in between, technically, one at a time. Syntax inspired by connevts from rule 14, in target language is brackets, and inside is the set, that returns a set of events.
- Previous rules used: Rule 4, defined on page 36, of the robochart reference manual, $allEvents(c : Context) : Set(Event)$.
- in TRule environment, latex, cannot make new lines, in second argument of environment, goes off edge in Rule 6, chansplit, metafunction declaration. Will fix, just noting.
- Just using c.name for name of process, what should it be, fully qualified name, to link to other semantic components, or will the process be renamed in circus?
- The only semantic rule we will have to implement, that is not here, in epsilon, is *allEvents*, Rule 4.
- As the RC semantics, we assume the existence of $eventId(e : Event)$, unique identifiers for the name of the event.
- for all, needs to be in order, it is a set of events, needs to be ordered. the implementation is ordered, we need to have an order, defined by the user, replicated according to an ordering function. That can be implemented, assume existence of *order* function, implemented by the implemetnation as a list.

Rule 1. Semantics of ANNs
$[\![\mathsf{c} : \mathsf{ANNController}]\!]_{\mathcal{ANN}} : \mathsf{Program} =$

$\mathsf{ANNChannelDecl(c)}$
$\mathsf{ANNProc(c)}$

Rule 2. Function ANNChannelDecl
$\mathsf{ANNChannelDecl(c : ANNController) : Program} =$

$\mathsf{ANNChannels(c, layerNo(c), lastLayerS)}$
channel *terminate*
channelset $ANNHiddenEvts == \{\!|\ \mathsf{hiddenEvts}\ |\!\}$
where
$\mathsf{hiddenEvts} = \{\mathsf{l}, \mathsf{n} : \mathbb{N} \mid (0 < \mathsf{l} < \mathsf{layerNo}) \land \mathsf{n} \in 1 \mathinner{..} \mathsf{layerSize(l)} \bullet \mathsf{layerRes(l, n)}\}$
$\mathsf{lastLayerS} = \mathsf{last}\ ((\mathsf{list}^{\sim})\mathsf{c.annparameters.layerstructure})$

- going to assume the existence of an $order(s : \mathbb{P}\, Event) : \mathrm{seq}\, Event$ function, that returns a sequence of events, of the same size of the set of events, can be implemented by lists, in Eclipse, by ordered, containement lists.

HELPER RULES:

3 AnglePIDANN Circus Program

3.1 Preliminary Material

3.2 Process Definition

4 Notes

Differences to the CSP semantics, where the Circus actions representing the CSP processes differ. We have proved, in FDR, for the *AnglePIDANN* and *AnglePIDANN*2 examples, and for a binarised version of *AnglePIDANN*, that our Circus semantics are equivalent, in the traces model, to the CSP semantics.

- channels are renamed, no longer use indexed channels, the indexed channel abstraction. There are multiple cases, on each process, with a guard and each process is chained together by external choice, such that only one process should not evaluate to $STOP$, the rest are $STOP \,\square\, P$, which evaluates to P.

- ANNHiddenEvts defined constructively, not all those events without the inputs and outputs.

Rule 3. Function ANNChannels
$\underline{\mathsf{ANNChannels(c : ANNController, l, n : \mathbb{N}) : Program =}}$

if $(\mathbf{l == 0}) \wedge (\mathbf{n == 1})$
then
    channel $\mathsf{layerRes(l, n)}$
else
    channel $\mathsf{layerRes(1, n)}$
    $\mathsf{if(l \neq 0)}$
        $\mathsf{NodeOutChannels(l, n, \# c.annparameters.layerstructure)}$
    $\mathsf{if(n > 1)}$
        $\mathsf{ANNChannels(l, (n-1))}$
    else
        $\mathsf{ANNChannels((l-1), LStructure(c, (l-1))}$

Rule 4. Function NodeOutChannels
$\underline{\mathsf{NodeOutChannels(l, n, i : \mathbb{N}) : Program =}}$

if $(\mathbf{i == 1})$
then
    channel $\mathsf{nodeOut(l, n, i)}$
else
    channel $\mathsf{nodeOut(1, n, i)}$
    $\mathsf{NodeOutChannels(l, n, (i-1))}$

Rule 5. Function ANNProc
$\underline{\mathsf{ANNProc(c : ANNController) : ProcDecl =}}$

process $\mathsf{c.name} \mathrel{\widehat=}$
    begin
    $\mathit{Collator} \mathrel{\widehat=} l, n, i : \mathbb{N};\ \mathit{sum} : \mathit{Value} \bullet\ \mathsf{Collator(c)}$
    $\mathit{NodeIn} \mathrel{\widehat=} l, n, i : \mathbb{N} \bullet\ \mathsf{NodeIn(c)}$
    $\mathit{Node} \mathrel{\widehat=} l, n, \mathit{inpSize} : \mathbb{N} \bullet\ \mathsf{Node(c)}$
    $\mathit{HiddenLayer} \mathrel{\widehat=} l, s, \mathit{inpSize} : \mathbb{N} \bullet\ \mathsf{HiddenLayer(c)}$
    $\mathit{HiddenLayers} \mathrel{\widehat=} \mathsf{HiddenLayers(c, layerNo(c) - 1)}$
    $\mathit{OutputLayer} \mathrel{\widehat=} \mathsf{OutputLayer(c)}$
    $\mathit{ANN} \mathrel{\widehat=} ((\mathit{HiddenLayers} [\![\mid \mathsf{IndexedLayerRes(layerNo(c) - 1)} \mid]\!] \mathit{OutputLayer})$
        $\setminus \mathit{ANNHiddenEvts})\ ;\ \mathit{ANN}$
    $\mathit{ANNRenamed} \mathrel{\widehat=} \mathsf{ANNRenamed(c)}$
    $\bullet\ \mathit{ANNRenamed}$
    end

Rule 6. Function Collator
$\underline{\mathsf{Collator(c : ANNController) : CSPAction}} =$

$$\Box\, \underline{\mathsf{l : 1 .. layerNo(c);\ n : 1 .. LStructure(c, l);\ i : 0 .. LStructure(c, (l-1))}}$$
$$\underline{\bullet}\left(l = \underset{.}{\mathsf{l}} \wedge n = \underline{\mathsf{n}} \wedge i = \underset{.}{\mathsf{i}}\right) \&$$
$\underline{\mathbf{if}(\mathbf{i == 0})}$
$\underline{\mathbf{then}}$
$\underline{\mathsf{layerRes(l, n)}}!relu(sum + (\underline{\mathsf{bias(c, l, n)}})) \longrightarrow \mathbf{Skip}$
$\underline{\mathbf{else}}$
$\underline{\mathsf{nodeOut(l, n, i)}}?x \longrightarrow Collator(l, n, (i-1), (sum + x))$

Rule 7. Function NodeIn
$\underline{\mathsf{NodeIn(c : ANNController) : CSPAction}} =$

$$\Box\, \underline{\mathsf{l : 1 .. layerNo(c);\ n : 1 .. LStructure(c, l);\ i : 1 .. LStructure(c, (l-1))}}$$
$$\underline{\bullet}\left(l = \underset{.}{\mathsf{l}} \wedge n = \underline{\mathsf{n}} \wedge i = \underset{.}{\mathsf{i}}\right) \&$$
$$\underline{\underline{\mathsf{layerRes(l, n)}}?\mathsf{x} \longrightarrow \underline{\mathsf{nodeOut(l, n, i)}}!(\mathsf{x} * \underline{\mathsf{weight(c, l, n, i)}}) \longrightarrow \mathbf{Skip}}$$

Rule 8. Function Node
$\underline{\mathsf{Node(c : ANNController) : CSPAction}} =$

$$\Box\, \underline{\mathsf{l : 1 .. layerNo(c);\ n : 1 .. LStructure(c, l)}}$$
$$\underline{\bullet}\left(l = \underset{.}{\mathsf{l}} \wedge n = \underline{\mathsf{n}}\right) \&\ ((\,|||\ i : 1 .. inpSize \bullet NodeIn(l, n, i))$$
$$[\![\ |\ \{\!|\ \underline{\mathsf{IndexedNodeOut(l, n)}}\ |\!\}\ |\ ]\!]$$
$$Collator(l, n, inpSize, 0) \setminus \{\!|\ \underline{\mathsf{IndexedNodeOut(l, n)}}\ |\!\})$$

Rule 9. Function HiddenLayer
$\underline{\mathsf{HiddenLayer(c : ANNController) : CSPAction}} =$

$$\Box\, \underline{\mathsf{l : 1 .. layerNo(c)}}$$
$$\underline{\bullet}\left(l = \underset{.}{\mathsf{l}}\right) \& ([\![\{\!|\ \underline{\mathsf{IndexedLayerRes(l)}}\ |\!\}\ ]\!]\ i : 1 .. s \bullet Node(l, i, inpSize))$$

Rule 10. Function HiddenLayers
$\underline{\mathsf{HiddenLayers}(\mathsf{c} : \mathsf{ANNController}, \mathsf{l} : \mathbb{N}) : \mathsf{CSPAction} =}$

$\underline{\textbf{if}(\mathbf{l == 1})}$
$\underline{\textbf{then}}$
$\quad (HiddenLayer(\mathsf{l}, \underline{\mathsf{LStructure}(\mathsf{l})}, \underline{\mathsf{LStructure}(\mathsf{l}-1)}))$
$\underline{\textbf{else}}$
$\quad (\underline{\mathsf{HiddenLayers}(\mathsf{c}, (\mathsf{l}-1)})$
$\quad [\![\mid \{\!|\ \underline{\mathsf{IndexedLayerRes}(\mathsf{c}, \mathsf{l}-1)}\ |\!\}\mid]\!]$
$\quad HiddenLayer(\mathsf{l}, \underline{\mathsf{LStructure}(\mathsf{l})}, \underline{\mathsf{LStructure}(\mathsf{l}-1)})$

Rule 11. Function OutputLayer
$\underline{\mathsf{OutputLayer}(\mathsf{c} : \mathsf{ANNController}) : \mathsf{CSPAction} =}$

$[\![\{\!|\ \underline{\mathsf{IndexedLayerRes}(\mathsf{layerNo}(\mathsf{c})-1)}\ |\!\}\,]\!]\ i : 1 \,..\, \underline{\mathsf{LStructure}(\mathsf{layerNo}(\mathsf{c}))} \bullet$
$\quad Node(l, i, \underline{\mathsf{LStructure}(\mathsf{layerNo}(\mathsf{c})-1)}))$

Rule 12. Function ANNRenamed
$\underline{\mathsf{ANNRenamed}(\mathsf{c} : \mathsf{ANNController}) : \mathsf{CSPAction}) =}$

$(ANN)\ \ [\underline{\mathsf{orderedLayerRes}} := \underline{\mathsf{eventList}}] \bigtriangleup terminate \longrightarrow \textbf{Skip}$
$\underline{\textbf{where}}$
$\underline{\mathsf{orderedLayerRes} =}$
$\quad \underline{\mathsf{order}(\{\mathsf{l} : \{0, \mathsf{layerNo}(\mathsf{c})\};\ \mathsf{n} : 1 \,..\, \mathsf{LStructure}(\mathsf{c}, \mathsf{l})} \bullet \underline{\mathsf{layerRes}(\mathsf{l}, \mathsf{n})\})}$
$\underline{\mathsf{eventList} = \mathsf{order}(\mathsf{allEvents}(\mathsf{c.annparameters.inputContext}))^\frown}$
$\quad \underline{\mathsf{order}(\mathsf{allEvents}(\mathsf{c.annparameters.outputContext}))}$

Rule 13. Function LStructure
$\underline{\mathsf{LStructure}(\mathsf{c} : \mathsf{ANNController}, \mathsf{i} : \mathbb{N}) : \mathbb{N} =}$

$\underline{\textbf{if}(\mathbf{i == 0})}$
$\underline{\textbf{then}}$
$\quad \underline{\#\,\mathsf{allEvents}(\mathsf{c.annparameters.inputContext})}$
$\underline{\textbf{else}}$
$\quad \underline{((\mathsf{list}^\sim)\mathsf{c.annparameters.layerstructure})\ \mathsf{i}}$

Rule 14. layerRes Channel
$\underline{\mathsf{layerRes}(\mathsf{l} : \mathbb{N}, \mathsf{n} : \mathbb{N}) : \mathsf{CSExp} =}$

$layerRes\underline{\mathsf{ln}} : Value$

Rule 15. Function IndexedLayerRes
$\underline{\mathsf{IndexedLayerRes}(\mathsf{l} : \mathbb{N}) : \mathsf{CSExp} =}$

$\{\!| \underline{\{\mathsf{n} : 1\,..\,\mathsf{LStructure}(\mathsf{l}) \bullet \mathsf{layerRes}(\mathsf{l}, \mathsf{n})\}} |\!\}$

Rule 16. nodeOut Channel
$\underline{\mathsf{nodeOut}(\mathsf{l} : \mathbb{N}, \mathsf{n} : \mathbb{N}, \mathsf{i} : \mathbb{N}) : \mathsf{CSExp} =}$

$nodeOut\underline{\mathsf{lni}} : Value$

Rule 17. Function layerNo (number of layers)
$\underline{\mathsf{layerNo}(\mathsf{c} : \mathsf{ANNController}) : \mathbb{N} =}$

$\underline{\#((\mathsf{list}^{\sim})\ \mathsf{c.annparameters.layerstructure})}$

Rule 18. Function weight
$\underline{\mathsf{weight}(\mathsf{c} : \mathsf{ANNController};\ \mathsf{l}, \mathsf{n}, \mathsf{i} : \mathbb{N}) : \mathsf{Value} =}$

$\underline{((\mathsf{tensor}^{\sim})\ \mathsf{c.annparameters.weights})\ \mathsf{l\ n\ i}}$

Rule 19. Function bias
$\underline{\mathsf{bias}(\mathsf{c} : \mathsf{ANNController};\ \mathsf{l}, \mathsf{n} : \mathbb{N}) : \mathsf{Value} =}$

$\underline{((\mathsf{matrix}^{\sim})\ \mathsf{c.annparameters.biases})\ \mathsf{l\ n}}$

Rule 20. Function AllNodeOut
$\underline{\mathsf{AllNodeOut}(\mathsf{c} : \mathsf{ANNController}) : \mathsf{CSExp} =}$

$\{\!| \underline{\{\mathsf{l} : 1\,..\,\mathsf{layerNo}(\mathsf{c});\ \mathsf{n} : 1\,..\,\mathsf{LStructure}(\mathsf{c}, \mathsf{l});\ \mathsf{i} : 1\,..\,\mathsf{LStructure}(\mathsf{c}, (\mathsf{l} - 1)) \bullet}$
$\quad \underline{\mathsf{nodeOut}(\mathsf{l}, \mathsf{n}, \mathsf{i})\}} |\!\}$

Rule 21. Function IndexedNodeOut
$\underline{\mathsf{IndexedNodeOut}(\mathsf{c} : \mathsf{ANNController}, \mathsf{l}, \mathsf{n} : \mathbb{N}) : \mathsf{CSExp} =}$

$\{\!| \underline{\{\mathsf{i} : 1\,..\,\mathsf{LStructure}(\mathsf{c}, \mathsf{l} - 1) \bullet \mathsf{nodeOut}(\mathsf{l}, \mathsf{n}, \mathsf{i})\}} |\!\}$

- We are not hiding all of node out, like we do in the original, because unnnecessary, and in the meta-language, we have to define it anyway, $\{\!| \mathit{nodeOut}.1.1 |\!\}$ we have to define anyway, so its cleaner, to define we synchronise on that, then hide just that.

- parallel synchronisation, without the variable sets, needs | characters, from the circus guide, it should not, but it does for us: $\llbracket \mid \{\!| \mathit{layerRes}11 |\!\} \mid \rrbracket$. $\llbracket \mathit{layerRes}11 \rrbracket$ does not compile, when it says this is valid syntax.

- No longer using replicated alphabetsied parallel in *HiddenLayers*, we are now using multiple generalised parallel in *HiddenLayers*.

Issues or Questions:

- Stateless, so we omit the state reserved word, allowed in CZT, but Marcel's BNF seem to imply it is always required.

- In this document, I am using the Circus latex style, not the csp or CZT, so we are using circinterrupt instead of interrupt for CSP interrupt, but both produce the same symbol.

- We use binarised parameters, and the *sign* activation function instead of *relu*, for validation of our Circus programs.

- We use the roboworld 2d toolkit, we only really need the real type, and the decimal point definition, in CZT, but we do need this declaration, otherwise the process would be very ugly, but this is required as well as the standard circus toolkit, to write the circus programs in CZT.

5 ANN Circus semantics in the Circus meta-model

5.1 Important Classes in metamodel

- Term (abstract class, represents a term).

- Para -¿ Term (abstract class, represents a paragraph).

- Types of Para: AxPara, ActionPara, ChannelPara, ChannelSetPara, ConjPara, FreePara, GivenPara, NameSetPara, ProcessPara, etc.

- Sect, is a Term, abstract class.

- Concrete subclasses of Sect, ZSect, that is it, just ZSect.

- ZSect -¿ Sect (Z section, has name: EString, paraList: ParaList, parents: Parent).

- ParaList, abstract class, list of paragraphs, ZParaList, paras, ZParaList is a concrete type of ParaList.

- ZParaList, concrete ParaList, list of, paras: Para.
- Para,
- ConstDecl, has name: ZName, and expr: Expr
- Expresssions, Expr, types of expression:
- Expr, is a term, an abstract class.
- Concrete instantiations: BasicChannelSetExpr, BindExpr, CondExpr, NumExpr, RefExpr, SigmaExpr, SchExpr, RefExpr,
- RefExpr
- SchTExt, ZSchText, schema text.
- ZName, is word, id, operatorName, strokesLsit, strokesList, list of Z strokes used.
- word is the name of ZName, an EString.
- id, 8571, just a number in CZT. real is

Notes from CZT API, https://czt.sourceforge.net/corejava/corejava-z/apidocs/index.html

- Stroke, is a Term, an abstract Stroke,
- Stroke, ?,

This is the AST,

5.2 Channel declarations, and the ReLU declaration

The CZT Circus AST, representation of our example, AST for our example:

- Horizontal Definition Paragraph (
- Schema text "Value"
- List of declarations "Value"
- Constant declaration "Value" (has a name and a reference expression) [ConstDecl in EMF]
- name, then a reference expression, has name,
- reference expression "real", [RefExpr in EMF]
- reference expression, has a list of expressions, [expression list].
-

Name of the constant declaration, is "Value", name of the reference expression, "real", list of expressions, no Z strokes,

5.3 Mechanisation Notes (EOL)

Mechanisation, in CZT Circus API, of the various BNF rules that we refer to, is:

- Program: multiple circus paragraphs, in CZT AST, this is: List of Paragraphs, in Tool, it is ZParaList, in Circus AST, you can put Circus and Z paragraphs in this, it is just a list of Paras, which can be either.
- ProcessPara, is ProcDecl, potentially, defining a process paragraph.
- We are using, in EOL, no c : ANNController, that is the self, that is the context of the operation, all the semantics are operations on ANNController objects.
- CircusProcess, absrtact, of BasicProcess, with parameters, mainAction, ontheFlyParagraphs, paragraph lists, Axiom paragraphs, state para. State paragraph list. Them has local paragraphs.
- ProcessPara, has a CircusProcess, a namelist, a name, and if it is a basic process or not.
- The AnglePIDANN, overall, is a process pargraph, not a circusprocess, then the basic process, then basic process has a list of paragraphs.
- Then, each is an action paragraph, each CSP process,
- Treat the $Value == \mathbb{R}$, horizontal definition paragraph, as Part of the Toolkit, as imported in CZT.
- Not using explicit paragraph lists, generating a document with just using, the actual paragraphs. They aren't grouped in the CZT file anyway, it was grouped, in the metalanguge, in the BNF, more just to show, to describe what they are.
- We don't have a section, that would have a list of paragraphs, we just have the list of paragraphs, doesn't matter. Still automatically generates them.
- Do we need the explicit import? and Section header? in the M2M? Does that exist yet?
- Each channel is declared in its own channel paragraph, one channel paragraph per channel declaration.
- layerRes and nodeOut are declared as STRINGS, not channels, as easier just to get the names, then call the functions to get the channel paragraphs, and declarations, from other places.
- Process Paragraph, top level, name "AnglePIDANN". Basic Process, then has list of paragraphs, aciton paragraphs, then horizontal definition paragraph.

- Axiomatic description paragraph relu. That can be in the toolkit as well.
- createProcessPara, needs a Z!CircusProcess, sets the circusprocess, to that, name: and is BasicProcess, isBasicProc, true,
- createParallelProcess, name, left, and right. cs_name, reference to channelset name, creates a channel set, with the reference expression, of cs_name.
- Can also, createParallelProcess, with $channel_names$ as a set, left, and right.
- createCallProcess, call expression,
- createBasicProcess, mainActionName, sets the main action.
- give a sequence of
- the basic process, paragraph lists, is the list.
- paragraphs, is the sequence.
- From a Z!Para, can create a basic process, with a main action name, this is a Z!BasicProcess.
- create action, createPrefixingAction, createAction1.
- mechanisation of process, top level: Process Paragraph, then has a name, the process paragraphs name is "AnglePIDANN", the name of the RC component. That is just the anglepidann.name, it does work.
- then has a basic process, then in this basic process, has a pargraph list, then it has ACTION PARAGRAPHS, then a main action, then a horizontal definition paragraph, default?
- Fang's mechanisation, find process paragraphs, then basic processes, and action paragraphs.
- createProcessPara(name: String, isBasicProc: Boolean), context of a CircusProcess, creates a ProcessPara. sets the name to name, and isBasicProcess,
- createCallProcess(), creates a Ref Expression, a process that calls a reference, a refernece expression.
- createParallelProcess,
- createActionPara(), Circus Action, returns an ActionPara, takes a CircusAction, creates an action paragraph. with the circus action = to self.
- createCircusAction,
- how the events are represented in memory,

- Events are not ordered, in EMF RoboChart models, based on when users, I saw that somewhere they were? But not in EOL itself, not ordered.
- ordered is FALSE on events, saw in representation, in parts of xtext code, not in EMF, not an ordered.
- ORDERING WORKS, NOT SURE WHY, SAYS UNORDERED, IN EMF, BUT IT SEEMS TO WORK, EVEN IN INTERFACES. MULTIPLE, ONE AT A TIME, SURELY.

A AnglePIDANN2 Circus Program

Used to make an example with more than one hidden layer, and more than one layer per node.

B CSP Semantics Sketch

$$Value == \mathbb{R}$$

$\textbf{channel}\ layerRes01 : Value$
$\textbf{channel}\ layerRes02 : Value$
$\textbf{channel}\ layerRes11 : Value$
$\textbf{channel}\ layerRes21 : Value$
$\textbf{channel}\ nodeOut111 : Value$
$\textbf{channel}\ nodeOut112 : Value$
$\textbf{channel}\ nodeOut211 : Value$
$\textbf{channel}\ terminate$

$\textbf{channel}\ adiff\_in : \mathbb{R}$
$\textbf{channel}\ anewError\_in : \mathbb{R}$
$\textbf{channel}\ angleOutputE\_out : \mathbb{R}$

$\textbf{channelset}\ ANNHiddenEvts == \{\!| layerRes11 |\!\}$

DON'T NEED THIS, for the meta-language.

$$\begin{array}{|l}
relu : Value \to Value \\
\hline
\forall x : Value \bullet \\
\quad (x < 0 \Rightarrow (x, 0) \in relu) \land \\
(x \geq 0 \Rightarrow (x, x) \in relu)
\end{array}$$

Figure 1: The preliminary Circus paragraphs, for the *AnglePIDANN* example.

$$
\begin{array}{l}
\textbf{process}\ \ \mathit{AnglePIDANN} \mathrel{\widehat=} \\
\quad \textbf{begin} \\
\quad \mathit{Collator} \mathrel{\widehat=} l, n, i : \mathbb{N};\ \ \mathit{sum} : \mathit{Value} \bullet \\
\qquad \big(l = 1 \land n = 1 \land i = 0\big) \mathbin{\&} \mathit{layerRes}11\,!(\mathit{relu}(\mathit{sum} + (0.125424))) \longrightarrow \textbf{Skip} \\
\qquad \Box\,\big(l = 1 \land n = 1 \land i = 1\big) \mathbin{\&} \mathit{nodeOut}111\,?x \longrightarrow \mathit{Collator}(l, n, (i - 1), (\mathit{sum} + x)) \\
\qquad \Box\,\big(l = 1 \land n = 1 \land i = 2\big) \mathbin{\&} \mathit{nodeOut}112\,?x \longrightarrow \mathit{Collator}(l, n, (i - 1), (\mathit{sum} + x)) \\
\qquad \Box\,\big(l = 2 \land n = 1 \land i = 0\big) \mathbin{\&} \mathit{layerRes}21\,!(\mathit{relu}(\mathit{sum} + (-\,0.107753))) \longrightarrow \textbf{Skip} \\
\qquad \Box\,\big(l = 2 \land n = 1 \land i = 1\big) \mathbin{\&} \mathit{nodeOut}211\,?x \longrightarrow \mathit{Collator}(l, n, (i - 1), (\mathit{sum} + x)) \\
\quad \mathit{NodeIn} \mathrel{\widehat=} l, n, i : \mathbb{N} \bullet \\
\qquad \big(l = 1 \land n = 1 \land i = 1\big) \mathbin{\&} \mathit{layerRes}01\,?x \longrightarrow \mathit{nodeOut}111\,!(x * (1.22838)) \longrightarrow \textbf{Skip} \\
\qquad \Box\,\big(l = 1 \land n = 1 \land i = 2\big) \mathbin{\&} \mathit{layerRes}02\,?x \longrightarrow \mathit{nodeOut}112\,!(x * (0.132874)) \longrightarrow \textbf{Skip} \\
\qquad \Box\,\big(l = 2 \land n = 1 \land i = 1\big) \mathbin{\&} \mathit{layerRes}11\,?x \longrightarrow \mathit{nodeOut}211\,!(x * (0.744636)) \longrightarrow \textbf{Skip} \\
\quad \mathit{Node} \mathrel{\widehat=} l, n, \mathit{inpSize} : \mathbb{N} \bullet \\
\qquad \big(l = 1 \land n = 1\big) \mathbin{\&}\ \ ((\,|||\ i : 1 \mathinner{..} \mathit{inpSize} \bullet \mathit{NodeIn}(l, n, i)) \\
\qquad\qquad\qquad\qquad \llbracket \mid \{\!|\, \mathit{nodeOut}111, \mathit{nodeOut}112 \,|\!\} \mid \rrbracket \\
\qquad\qquad\qquad\qquad \mathit{Collator}(l, n, \mathit{inpSize}, 0) \setminus \{\!|\, \mathit{nodeOut}111, \mathit{nodeOut}112 \,|\!\}) \\
\qquad \Box\,\big(l = 2 \land n = 1\big) \mathbin{\&}\ \ ((\,|||\ i : 1 \mathinner{..} \mathit{inpSize} \bullet \mathit{NodeIn}(l, n, i)) \\
\qquad\qquad\qquad\qquad \llbracket \mid \{\!|\, \mathit{nodeOut}211 \,|\!\} \mid \rrbracket \\
\qquad\qquad\qquad\qquad \mathit{Collator}(l, n, \mathit{inpSize}, 0) \setminus \{\!|\, \mathit{nodeOut}211 \,|\!\}) \\
\quad \mathit{HiddenLayer} \mathrel{\widehat=} l, s, \mathit{inpSize} : \mathbb{N} \bullet \\
\qquad (\llbracket \{\!|\, \mathit{layerRes}01, \mathit{layerRes}02 \,|\!\} \rrbracket\ i : 1 \mathinner{..} s \bullet \mathit{Node}(l, i, \mathit{inpSize})) \\
\quad \mathit{HiddenLayers} \mathrel{\widehat=} \\
\qquad \mathit{HiddenLayer}(1, 1, 2) \\
\quad \mathit{OutputLayer} \mathrel{\widehat=} \\
\qquad \llbracket \{\!|\, \mathit{layerRes}11 \,|\!\} \rrbracket\ i : 1 \mathinner{..} 1 \bullet \mathit{Node}(2, i, 1) \\
\quad \mathit{ANN} \mathrel{\widehat=} \\
\qquad ((\mathit{HiddenLayers} \llbracket \mid \{\!|\, \mathit{layerRes}11 \,|\!\} \mid \rrbracket \mathit{OutputLayer}) \setminus \mathit{ANNHiddenEvts})\ ;\ \ \mathit{ANN} \\
\quad \mathit{ANNRenamed} \mathrel{\widehat=} \\
\qquad (\mathit{ANN})\ \ [\mathit{layerRes}01, \mathit{layerRes}02, \mathit{layerRes}21 := \\
\qquad\qquad\qquad \mathit{anewError\_in}, \mathit{adiff\_in}, \mathit{angleOutputE\_out}] \\
\qquad\qquad \triangle\ \mathit{terminate} \longrightarrow \textbf{Skip} \\
\quad \bullet\ \mathit{ANNRenamed} \\
\quad \textbf{end}
\end{array}
$$

Figure 2: *AnglePIDANN* example, in Circus

process $AnglePIDANN2 \mathrel{\widehat=}$ **begin**

$Collator \mathrel{\widehat=} l, n, i : \mathbb{N};\ sum : Value \bullet$

$(l = 1 \land n = 1 \land i = 0)\ \&\ layerRes11\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 1 \land n = 1 \land i = 1)\ \&\ nodeOut111\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 1 \land n = 1 \land i = 2)\ \&\ nodeOut112\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 1 \land n = 2 \land i = 0)\ \&\ layerRes12\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 1 \land n = 2 \land i = 1)\ \&\ nodeOut121\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 1 \land n = 2 \land i = 2)\ \&\ nodeOut122\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 1 \land n = 3 \land i = 0)\ \&\ layerRes13\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 1 \land n = 3 \land i = 1)\ \&\ nodeOut131\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 1 \land n = 3 \land i = 2)\ \&\ nodeOut132\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 2 \land n = 1 \land i = 0)\ \&\ layerRes21\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 2 \land n = 1 \land i = 1)\ \&\ nodeOut211\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 2 \land n = 1 \land i = 2)\ \&\ nodeOut212\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 2 \land n = 1 \land i = 3)\ \&\ nodeOut213\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 3 \land n = 1 \land i = 0)\ \&\ layerRes31\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 3 \land n = 1 \land i = 1)\ \&\ nodeOut311\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 4 \land n = 1 \land i = 0)\ \&\ layerRes41\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 4 \land n = 1 \land i = 1)\ \&\ nodeOut411\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$\Box(l = 4 \land n = 2 \land i = 0)\ \&\ layerRes42\,!(sign(sum + (0))) \longrightarrow \mathbf{Skip}$

$\Box(l = 4 \land n = 2 \land i = 1)\ \&\ nodeOut421\,?x \longrightarrow Collator(l, n, (i - 1), (sum + x))$

$NodeIn \mathrel{\widehat=} l, n, i : \mathbb{N} \bullet$

$(l = 1 \land n = 1 \land i = 1)\ \&\ layerRes01\,?x \longrightarrow nodeOut111\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 1 \land n = 1 \land i = 2)\ \&\ layerRes02\,?x \longrightarrow nodeOut112\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 1 \land n = 2 \land i = 1)\ \&\ layerRes01\,?x \longrightarrow nodeOut121\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 1 \land n = 2 \land i = 2)\ \&\ layerRes02\,?x \longrightarrow nodeOut122\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 1 \land n = 3 \land i = 1)\ \&\ layerRes01\,?x \longrightarrow nodeOut131\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 1 \land n = 3 \land i = 2)\ \&\ layerRes02\,?x \longrightarrow nodeOut132\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box$

$(l = 2 \land n = 1 \land i = 1)\ \&\ layerRes11\,?x \longrightarrow nodeOut211\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box(l = 2 \land n = 1 \land i = 2)\ \&\ layerRes12\,?x \longrightarrow nodeOut212\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box(l = 2 \land n = 1 \land i = 3)\ \&\ layerRes13\,?x \longrightarrow nodeOut213\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box(l = 3 \land n = 1 \land i = 1)\ \&\ layerRes21\,?x \longrightarrow nodeOut311\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box(l = 4 \land n = 1 \land i = 1)\ \&\ layerRes31\,?x \longrightarrow nodeOut411\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$\Box(l = 4 \land n = 2 \land i = 1)\ \&\ layerRes31\,?x \longrightarrow nodeOut421\,!(x * (1)) \longrightarrow \mathbf{Skip}$

$Node \mathrel{\widehat=} l, n, inpSize : \mathbb{N} \bullet$

$(l = 1 \land n = 1)\ \&\ ((\,|||\ i : 1 \mathinner{..} inpSize \bullet NodeIn(l, n, i))$

$[\![\ |\ \{\!|\ nodeOut111, nodeOut112\ |\!\}\ |\ ]\!]$

$Collator(l, n, inpSize, 0) \setminus \{\!|\ nodeOut111, nodeOut112\ |\!\}$

$)$

$\Box$

$(l = 1 \land n = 2)\ \&\ ((\,|||\ i : 1 \mathinner{..} inpSize \bullet NodeIn(l, n, i))$

$[\![\ |\ \{\!|\ nodeOut121, nodeOut122\ |\!\}\ |\ ]\!]$

$Collator(l, n, inpSize, 0) \setminus \{\!|\ nodeOut121, nodeOut122\ |\!\}$

$)$

$\Box$

$(l = 1 \land n = 3)\ \&\ ((\,|||\ i : 1 \mathinner{..} inpSize \bullet NodeIn(l, n, i))$

$$
\begin{array}{l}
\mathit{ANNRenamed} = \mathit{ANN} \bigtriangleup \mathit{end} \longrightarrow \mathit{SKIP} \\
\mathit{ANN} = \\
\quad ((\mathit{HiddenLayers} \parallel [\{\mid \mathit{layerRes}.(\mathit{layerNo} - 1) \mid\}] \mathit{OutputLayer}) \setminus \mathit{ANNHiddenEvts}) \\
\quad ;\ \mathit{ANN} \\
\mathit{ANNHiddenEvts} = \Sigma \setminus \{\mid \mathit{layerRes}.0, \mathit{layerRes}.\mathit{layerNo}, \mathit{end} \mid\} \\
\mathit{HiddenLayers} = \parallel i : 1 \mathinner{..} \mathit{layerNo} - 1 \bullet [\ \{\mid \mathit{layerRes}.(i-1), \mathit{layerRes}.i \mid\}\ ] \\
\qquad\qquad \mathit{HiddenLayer}(i, \mathit{layerSize}(i), \mathit{layerSize}(i-1)) \\
\mathit{HiddenLayer}(l, s, \mathit{inpSize}) = \parallel i : 1 \mathinner{..} s \bullet [\{\mid \mathit{layerRes}.(l-1) \mid\}] \mathit{Node}(l, i, \mathit{inpSize}) \\
\mathit{Node}(l, n, \mathit{inpSize}) = \\
\quad (\ (\mathop{|||}\ i : 1 \mathinner{..} \mathit{inpSize} \bullet \mathit{NodeIn}(l, n, i)) \\
\qquad\quad \parallel [\ \{\mid \mathit{nodeOut}.l.n \mid\}\ ] \\
\qquad \mathit{Collator}(l, n, \mathit{inpSize})\ ) \setminus \{\mid \mathit{nodeOut} \mid\} \\
\mathit{NodeIn}(l, n, i) = \mathit{layerRes}.(l-1).n?x \longrightarrow \mathit{nodeOut}.l.n.i!(x * \mathit{weight}) \longrightarrow \mathbf{Skip} \\
\mathit{Collator}(l, n, \mathit{inpSize}) = \mathbf{let} \\
\qquad\qquad C(l, n, 0, \mathit{sum}) = \mathit{layerRes}.l.n!(\mathit{ReLU}(\mathit{sum} + \mathit{bias})) \longrightarrow \mathbf{Skip} \\
\qquad\qquad C(l, n, i, \mathit{sum}) = \mathit{nodeOut}.l.n.i?x \longrightarrow C(l, n, (i-1), (\mathit{sum} + x)) \\
\qquad \mathbf{within} \\
\qquad\qquad C(l, n, \mathit{inpSize}, 0) \\
\mathit{OutputLayer} = \parallel i : 1 \mathinner{..} \mathit{layerSize}(\mathit{layerNo}) \bullet\ [\ \{\mid \mathit{layerRes}.(\mathit{layerNo} - 1) \mid\}\ ] \\
\qquad\qquad \mathit{Node}(\mathit{layerNo}, i, \mathit{layerSize}(\mathit{layerNo} - 1))
\end{array}
$$

Figure 4: CSP ANN Semantic Pattern.