CPN Tools

CPN Tools is a tool for editing, simulating, and analysing coloured Petri nets. CPN Tools is originally developed by the CPN Group at Aarhus University, Denmark and now maintained by the IS group at Eindhoven University of Technology, Netherlands.

CPN Tools has over ten thousand licensees in 143 countries, and more than 600 of these are commercial licenses. CPN Tools is widely used and provides a mature platform for working with high-level Petri nets in both academic and industrial settings.

We provide exercises intended to get you started simulating an existing model, exercises to perform simple modifications to the model, debug the changes using simulation, and verify correctness of your modifications using state space analysis. These exercises are intended for the tool hands-on session on Friday, September 17.

We also provide larger projects intended to get you started modelling using coloured Petri nets yourself. These projects are intended for the CPN Tools hands-on session on Friday, September 24. We recommend browsing through the projects before Wil van der Aalst's lecture on Monday, September 20, to put the lecture into perspective.

CPN Tools has a user interface which takes a bit getting used to, so in the following we provide an introduction to the GUI of CPN Tools as well as information on getting started performing simple simulation and modification of CPN models. In addition to the written material you are currently reading, we have also produced a couple introduction videos intended to help you get started as fast as possible. We suggest watching the introduction videos and using the written material as a reference when working with the tool yourself. Get the videos at

> http://tinyurl.com/cpntools-intro or
> http://www.youtube.com/view_play_list?p=1F2555094072F2F8

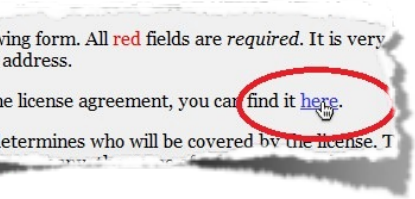
Michael Westergaard
August 2010
Eindhoven, Netherlands

Obtaining CPN Tools

To use CPN Tools, you need a license. You can either apply for a license on your own or use the credentials provided below. If you choose the latter option, skip the section on applying for a license, and jump directly to downloading CPN Tools.

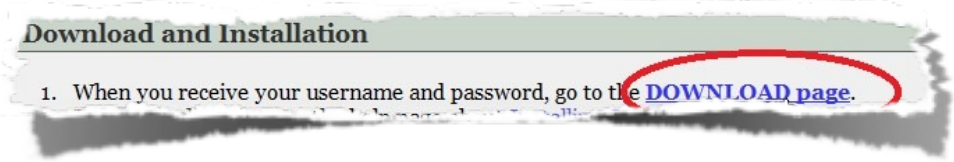
Applying for a CPN Tools License

- Go to http://www.cs.au.dk/CPNTools
- Click on *Apply for CPN Tools license* under the *Download and Installation* heading
- Fill in the form; fields marked in red are mandatory
  - Make sure you read the license agreement linked at the top and bottom of the page
  - Click on *Submit*
  - You receive an e-mail at your specified address containing download credentials

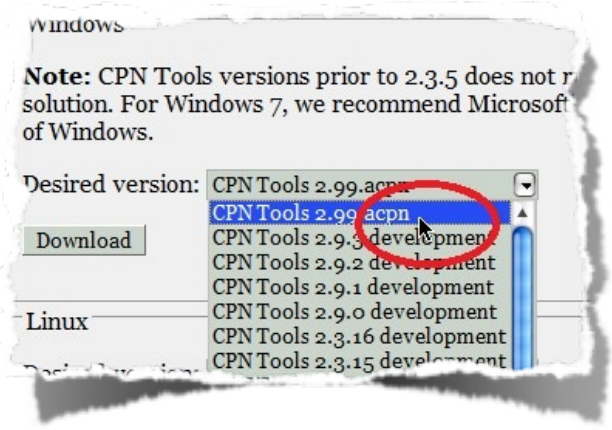


Downloading CPN Tools

- Go to http://www.cs.au.dk/CPNTools
- Click on *Download CPN Tools* under the *Download and Installation* heading
- Click on the *DOWNLOAD page* link under the *Download and Installation* heading
- Enter the username/passwords combination obtained above or use the combination
  username: acpn10 (lowercase A-C-P-N-one-zero)
  password: acpn10
- Click *Submit Query*
- Pick version *2.99.acpn* for Windows in the drop down menu



- **NOTE for Linux/Mac users**: we strongly recommend using the Windows version of CPN Tools under virtualization for your platform. Use the Linux version at your own risk! The tutors will **not** be able to help you get the Linux version running running the course!
- Click *Download* and install CPN Tools as a normal Windows application



CPN Tools Exercises

Friday, September 17

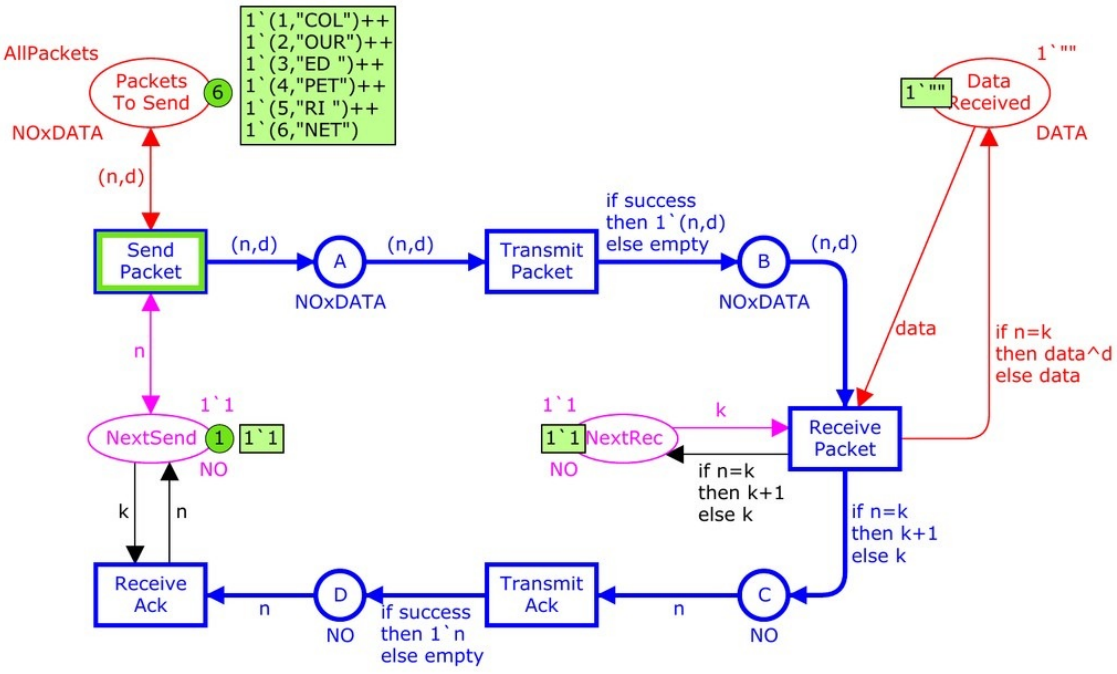
All of these exercises are from the CPN book (*Coloured Petri Nets - Modelling and Validation of Concurrent Systems*). The book homepage is at
http://www.cs.au.dk/CPnets/cpnbook
and there you can find the models you should use as basis for some of the exercises.

You can solve the exercises alone of in groups of 2-3 persons. You are not expected to have time to solve all the exercises.

Exercise 1: Interactive and Automatic Simulation

Goal: Get acquainted with CPN Tools and its simulation tools.
Expected time: 20 min.
Prerequisite: Sections or videos on CPN Tools user interface and simulation.

In this and the following exercises, we consider this simple stop and wait network protocol. Get the model from the above link to the CPN book homepage or directly by going to http://tinyurl.com/simpleprotocol.



Consider this simple protocol. Use the CPN simulator to perform interactive and automatic simulations of the CPN model to answer questions like: does the protocol terminate? is the the final state unique if it terminates? how is packet loss/ duplication/reordering handled by the protocol?

Exercise 2: Reception of Acknowledgements

Goal: Learn to do simple modifications to inscriptions of an existing model.
Expected time: 30 min.
Prerequisites: Exercise 1 and the section or video on editing. You may benefit from the SML Basis library (http://www.standardml.org/Basis), but this is not required.

Consider the simple protocol from exercise 1. When an acknowledgement is received by the sender, it updates the counter on NextSend according to the number contained in the acknowledgement. This implies that the counter on NextSend can be decreased when an "old" acknowledgement is received. Use the CPN simulator and interactive simulation to construct a scenario where this situation occurs.

Modify the CPN model such that the counter on NextSend will never be decreased. Use simulation to validate the modified model.

Exercise 3: Bounded Retransmission

Goal: Learn to do simple modifications to the net structure of an existing model.
Expected time: 30 min.
Prerequisites: Exercise 1 and the section or video on editing.

Consider the simple protocol from exercise 1. The CPN model specifies no upper bound on the number of times that a data packet can be transmitted. Modify the CPN model such there is a bound on the number of times that a packet can be retransmitted. Use simulation to validate the modified model.

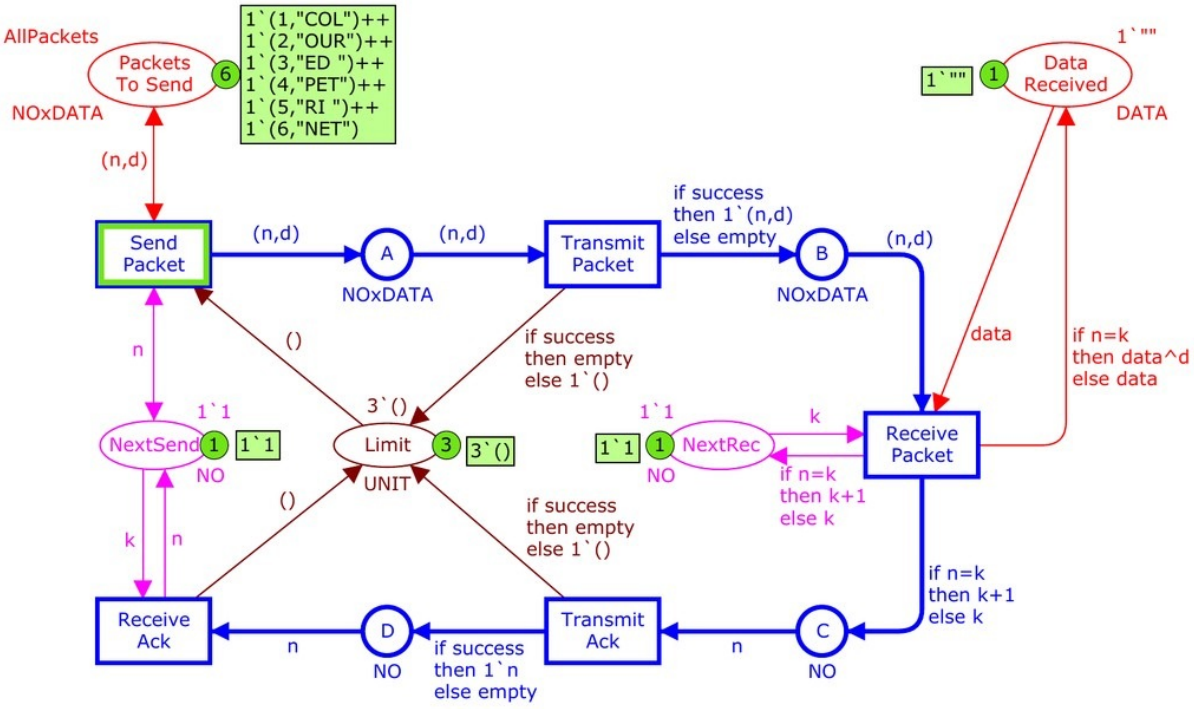
Exercise 4: Bounded Retransmission State-space Analysis

Goal: Use state space analysis to verify correctness of a model.
Expected time: 40 min.
Prerequisites: Exercise 3 and the video on state space analysis. You may also find the section on state space analysis in the CPN Tools documentation useful (http://tinyurl.com/statespace).

In this exercise, we consider this modified version of the simple protocol to limit the number of packets on the network at any time. Get the model from the above link to the CPN book homepage or directly by going to http://tinyurl.com/boundedprotocol (see a screenshot from the protocol on the next page).

Integrate your solution to exercise 3 into the modified CPN model of the simple protocol (or the other way around).

Investigate the correctness of this variant of the simple protocol. Does the protocol terminate? Is the result correct? Is it always possible to get to the correct end state? Is



it possible to make the protocol correct using bounded retransmission? Why/why not?

You should start by considering a small number of packets (2-3) to be transmitted and a small value of the retransmission limit (e.g., 2). The query functions *ListDeadMarkings* and *HomeSpace* might be useful. Investigate the protocol for different values of the protocol parameters.

Exercise 5: Go-Back-N Window Protocol

Goal: Making more substantial model changes.
Expected time: 45 min.
Prerequisites: Exercise 3.

The simple protocol is a stop-and-wait protocol: The sender keeps sending the same packet until the matching acknowledgement is received. In sliding window protocols, it is possible for the sender to transmit several packets to the receiver before receiving an acknowledgement, i.e., the sender has a sender window containing w data packets which is currently under transmission and for which acknowledgement have not yet been received.

Modify the CPN model such that it models a Go-Back-N Window Protocol. In a Go-Back-N protocol, the sender sends all data packets in the current window and then wait for acknowledgements. If no acknowledgement is received (within a certain amount of time), the data packets in the window are all retransmitted. It should only be necessary to modify the Sender part of the CPN model.

Use simulation to validate the correctness of the protocol.

Exercise 6: Selective Repeat Window Protocol

Goal: Making more substantial model changes.
Expected time: 45 min.
Prerequisites: Exercises 3 and 5.

If you are running out of time, it may be more interesting to skip to exercise 7 and get back to this afterwards.

The Go-Back-N Window Protocol introduces the concept of a sender window. The basic idea of the Selective Repeat Window Protocol is that there is also a receiver window that allows the receiver to buffer packets received out-of-order. Furthermore, the receiver can explicitly request retransmission of a "missing" packet from the receiver by sending a negative acknowledgement.

Modify the CPN model such that it specifies a Selective Repeat Window Protocol.

Use simulation to validate the correctness of the protocol.

Exercise 7: State-space Analysis of Window Protocol

Goal: Using state space analysis to find and fix errors in models.
Expected time: 45 min.
Prerequisites: Exercises 4 and 5/6.

In exercise 5 (or 6) you developed a CPN model specifying a sliding window protocol. Integrate your solution to exercise 5 (or 6) into the CPN model of the simple protocol with bounded network capacity from exercise 3 (or the other way around).

Use state space analysis to verify your sliding window protocol model. If you discover errors in your protocol, try to find a counter example and try also to fix any identified errors.

CPN Tools Projects

Friday, September 24

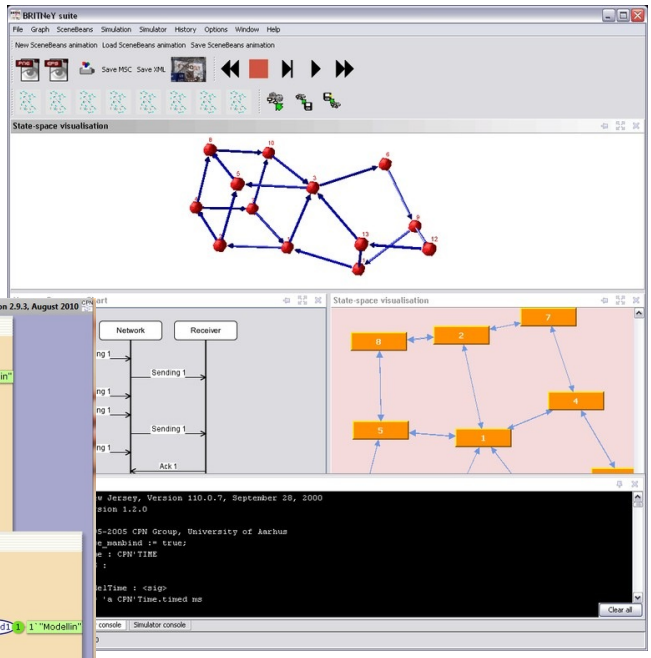
The purpose of the projects is to use CPN modelling to solve realistic problems. You are expected to build a model of a system and validate your design using either simulation or state space analysis. You can also visualise the execution of your model and use it to generate, e.g., sequence diagrams. We provide some project suggestions, but you are also welcome to define a project yourself as long as it is an interesting case. Ask Michael for approval if you wish to design a project yourself.

You are expected to be familiar with all the material on CPN Tools in his booklet and to have solved at least exercises 1, 2, 3, and 4 from last week (or 5/6 instead of 3 and 7 instead of 4). If you have not done so, you should do this before starting on projects.

You should solve one or more of the projects. You will not have time to do them all, and if you do a good job on just one, it is ok if you only get to do that. You should do the projects in groups of 2-3 persons. Doing the project alone is *not* advised!

Each of the projects deals with a set of tasks to get you started and a set of proposed tasks that provide interesting directions to proceed to dig deeper into the project. Use the proposed tasks as guidelines for what to do rather than a strict check list; they are just there to inspire you. Also, read through the list of proposed projects and choose the one that appeals the most to you instead of starting from the beginning.

During the day, I will give a couple brief demonstrations of CPN models used in industrial settings and how they have been used to solve practical problems.



Useful Resources

CPN-ML (inscription language)

- Overview at http://wiki.daimi.au.dk/cpntools-help/cpn_ml.wiki
- Standard ML Basis library at http://www.standardml.org/Basis/overview.html

Hierarchical models in CPN Tools

- Overview at http://wiki.daimi.au.dk/cpntools-help/working_with_hierarchical.wiki

Simulation-based performance analysis in CPN Tools

- Overview at http://wiki.daimi.au.dk/cpntools-help/performance_analysis.wiki
- Data collector monitors at http://wiki.daimi.au.dk/cpntools-help/data_collector_monitors.wiki
- Simulation replications at http://wiki.daimi.au.dk/cpntools-help/simulation_replications.wiki

State space analysis in CPN Tools

- Intro video at http://tinyurl.com/cpntools-intro
- Overview at http://wiki.daimi.au.dk/cpntools-help/use_state_space_tool.wiki
- Manual at http://wiki.daimi.au.dk/cpntools-help/manual.pdf.wiki

Visualisation using the BRITNeY Suite and CPN Tools

- BRITNeY Suite homepage at http://wiki.daimi.au.dk/britney
- Tutorial at http://britney.szm.com/en
- BRITNeY Suite examples at http://wiki.daimi.au.dk/britney/examples.wiki

Log generation for process mining

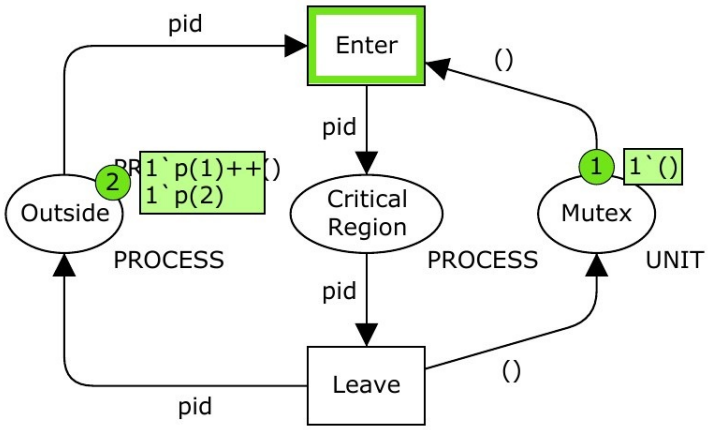
- Library for generating logs from CPN Tools for import into ProM at http://prom.win.tue.nl/research/wiki/tools/promcpn
- ProMimport 7 (you need to convert the output from the library using this prior to import into ProM 6) at http://prom.win.tue.nl/research/wiki/promimport

Project 1: Mutual Exclusion

Validation of parallel and distributed algorithms.
Focus: State space analysis.

Mutual exclusion algorithms are important in parallel and distributed settings where multiple participants intend for a shared resource, such as a piece of hardware or just a memory location inside a multi-threaded application.

Normally, we would model this like below in a CPN model, ignoring the actual implementation details.



This implementation assumes that we can at once query for and obtain the mutex or that "something" handles mutual exclusion for us. This exercise is about exploring various ways of doing so.

Task 1) Implement the above model in CPN Tools.

Task 2) Prove using state space analysis that the model works (consider what that means and how you can verify this using the state space tool).

Task 3) Implement Hyman's algorithm (below) for mutual exclusion. Check if the algorithm works.

```
init()
    flag = bool[2]
    k = 0
```

```
enter(int pid)
    flag[pid] = true
    while (k != pid)
        while (flag[1 - pid])
            skip
        k = pid
```

```
leave(int pid)
    flag[pid] = false
```

Task 4) Implement Peterson's algorithm (next page) for mutual exclusion. Check if the algorithm works.

```
init()
    flag = bool[2]
    k = 0
```

```
enter(int pid)
    flag[pid] = true
    k = 1 - pid
    while (flag[1 - pid] &
            turn != pid)
        skip
```

```
leave(int pid)
    flag[pid] = false
```

Proposed task 1) Implement a couple other mutex algorithms (Wikipedia has a list).

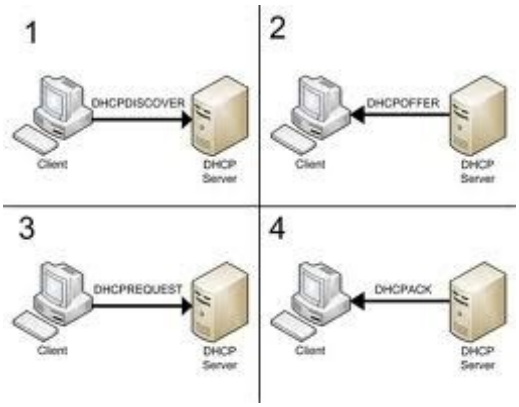
Proposed task 2) Implement a couple distributed mutex algorithms (e.g., central server, token ring, Lamport's distributed mutex algorithm).

Proposed task 3) Use simulation and the performance tool to figure out which algorithm performs the best (what does that mean? do you need to use time?).

Project 2: The DHCP Protocol

Modelling and verification of a real-life network protocol.
Focus: Modelling of network protocols.

The DHCP protocol is used to assign IP addresses and other information to computers when they start up without requiring manual configuration. The DHCP protocol is underlying the IT infrastructure almost everywhere, so it is crucial that it is working correctly.



This project is concerned with evaluation of the DHCP protocol.

Task 1) Get acquainted with the relevant parts of the DHCP protocol (RFC 2131; can be obtained at http://www.ietf.org/rfc/rfc2131.txt).

Task 2) Make a CPN model of relevant parts of DHCP. Think about which details to include and which to abstract away. did you find anything unclear/missing in the specification or maybe even contradictions?

Proposed task 1) State some correctness criteria for the DHCP protocol and verify them using state space analysis.

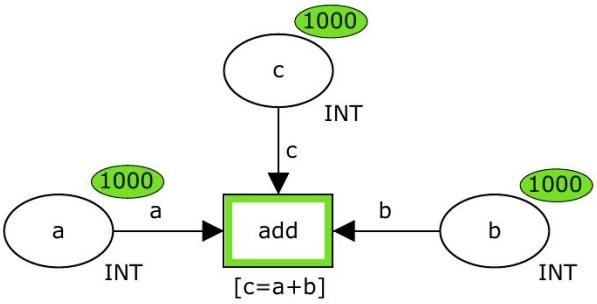
Proposed task 2) Use the BRITNeY Suite to generate message sequence charts from your model. Can you get your model to reproduce the message sequence charts from the RFC?

Proposed task 3) Use simulation based performance analysis to test the performance of the protocol. Do you need to make your model timed to do this?

Project 3: Enabling Calculation for CP-Nets

Petri nets for algorithm design.
Focus: Algorithms.

Enabling calculations for coloured Petri nets is far from easy if you want to go beyond the most naive approach (try all possible tokens). For an efficient computer tool, this is necessary, however. Consider this model:



The numbers in the circles represent the number of tokens (integers 0, ..., 999), so for this transition we have 1.000.000.000 possible bindings, but "only" 500.500 of these are actually enabled. Were we to seek randomly for an enabled binding, we would only have a one in approximately 2000 chance of finding a binding that is enabled. Even worse, if *c* contained 1000 tokens with values 2000, ..., 2999, we would have to search 10^9 possible bindings before concluding the transition is not enabled. Yet, CPN Tools immediately answers whether this transition is enabled. This exercise will investigate how CPN Tools calculates enabling of transitions.

Task 1) Create the above model in CPN Tools. You may benefit from knowing that

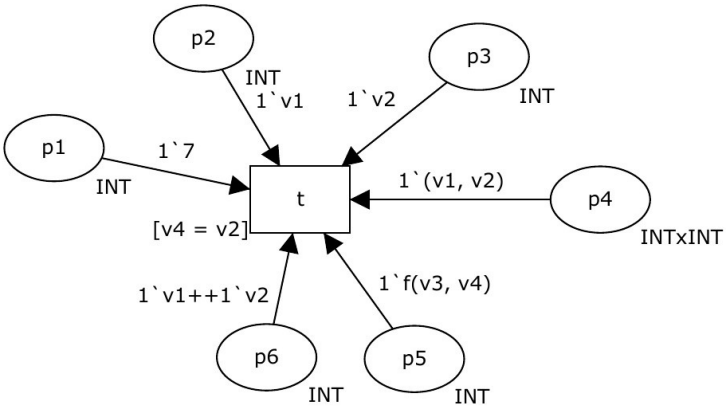
```
List.tabulate (1000, fn n => 2000 + n)
```

produces a multi-set consisting of the numbers 2000, ..., 2999. How would you calculate enabling for *add*? How would your algorithm be affected by the number of tokens on *a*, *b*, and *c* and whether *add* turns out to be enabled or not? How is CPN Tools affected?

As Task 1 illustrates, CPN Tools does not use the trivial algorithm for enabling calculation. CPN Tools can bind variables in 6 different ways.

Our task is to bind each variable to a value as efficiently as possible. We do this by analysing the expressions on arcs and the guard. In the following we will use the term *expression* to denote an expression where all variables are bound to a value (also known as a closed expression) and the term *pattern* to refer to one with unbound variables (also known as an open expression). We only allow patterns where we do not do any computation on unbound variables, i.e., *v1* is okay, but *v1 + 7* is not. We allow tuples of variables (actually anything which in SML is considered as a pattern).

Consider the example to the right. It contains 4 variables that need to be bound. *v1*, *v2*, and *v4* are of type *INT* and *v3* is of type *BOOL*. This model has (at least) the same problems as the previous if the places contain many tokens. We can bind variables in the following ways:



- On the arc from *p1* there is a constant expression. We only need to check that the value exists on *p1*.
- On the arcs from p2, p3, and p4 are patterns. The variables can be bound by picking a random token from the corresponding places.
- If we have already bound *v1* somewhere else (e.g., from the arc from *p2*), the arc from *p4* contains a pattern where a subset of the variables are bound. We can bind the remaining variables by using the known variables as a key and simply look up the possible values for the remaining variables in the tokens of the place.
- On the arc from *p5* there is a function with two arguments. We cannot consider this a pattern as we do computation on unbound variables. Instead we need to bind the variables from somewhere else and subsequently check if the result of the computation is available on the place.
- On the arc from *p6* is an expression which can be split into two, one for each addend.
- The guard can be used to bind either *v2* or *v4* if the other is known. If both are known, we can evaluate the guard to test if the transitions enabled.

Using these observations, we can find 6 ways to bind variables:

Operator	Purpose	Weight
Bp pat	Bind by randomly choosing token on place	100
Bk pat	Bind remaining by using bound variables as key	20
Bc pat	Bind variables by randomly picking values for variables of small colour sets (bool, enum, index, certain subsets)	1000
Ta exp	Assuming all variables are bound, test if the evaluated expression exists on the place	10
Bg pat = exp	In the guard: Bind variables in *pat* by evaluating *exp* (which has all variables bound)	1
Tg	In the guard: Test if *exp* (which has all variables bound) is true	2

Task 2) Use the rules from the table to find binding strategies for the add transition. Note that some rules can only be applied after applying others. What is the weight of the different binding sequences.

As illustrated, different binding strategies lead to different weights. Often the binding strategy with the lowest weight is the faster one. We therefore wish to find the best binding strategy, i.e., the one with the lowest weight. An elegant way to do that is to construct a Petri net where each place corresponds to a variable and each transition to a possible binding rule. Places contain a single token, a boolean representing whether the variable is bound or not, and each transition is connected to all places corresponding to variables it assumes are bound and all places corresponding to variables it binds. We also add a place indicating whether the guard is satisfied. A binding sequence is a sequence of transitions leading from the initial state, where all places obtain a single token *false*, to a state where all places contain a single token *true*.

Task 3) Construct such a Petri net for the *add* example. Your model should calculate the cost of a generated binding sequence (whenever a transition is executed, it becomes part of the binding sequence). Use state space analysis to figure out the best binding sequence. How does this correspond to the one you found in Task 1?

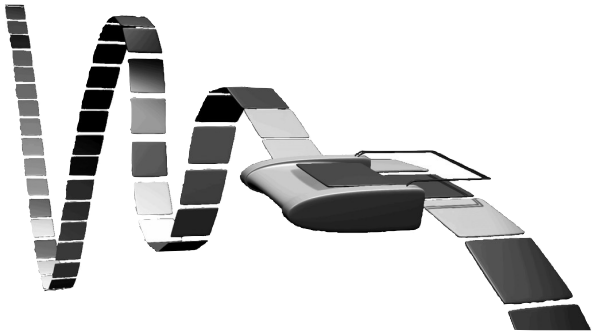
Task 4) Construct optimal binding sequences for all transitions in the simple protocol used last week.

Proposed task 1) In the example with t and p1, ..., p6, remove the guard. What happens to the binding sequences? How can this be exploited?

Proposed task 2) What limitations does this way of making bindings have? Can you come up with example transitions for which a binding cannot be found (test this in CPN Tools)?

Proposed task 3) We have only looked at single processor binding calculation here (and this is also what CPN Tools implements). Can the approach be extended to find the most efficient bindings on a multi-processor system? Does this yield a practical improvement (test your approach on the transitions from the simple protocol)?

Project 4: Turing Machines



Computability of coloured Petri nets.
Focus: Modelling techniques.

Turing machines are a model of computability which coincides with many other natural notions of computability (like Gödel's recursive functions or Chomsky's context-sensitive grammars) and is used as a standard benchmark of programming languages - if a languages is Turing complete computer scientists know it is as powerful as all other languages.

A Turing machine informally consists of

- a (infinite in one direction) **tape** containing cells each of which can contain a symbol from a given alphabet,
- a **head** that can read and write symbols as well as move one cell left or right,
- an **action table** mapping a state and a letter of the alphabet to a new letter, a new state, and a direction (left or right), meaning if the Turing machine is in a state and the head reads a letter, it should write the new letter, switch to the new state and move the head left or right on the tape, and
- a **state register** containing the current state.

You can get a formal definition from, e.g., Wikipedia.

Task 1) Implement a Turing machine as a CPN model. Your model should be able to execute any Turing machine by changing the state table.

Task 2) Run your model with a concrete Turing machine (e.g., one multiplying two numbers given on the tape as input).

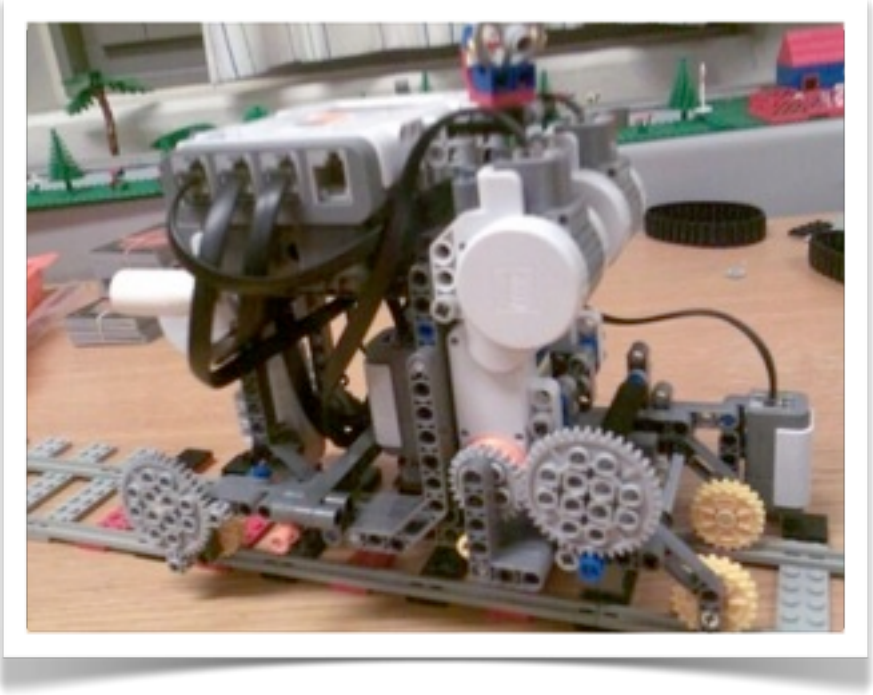
Task 3) A common problem with modelling with high-level Petri nets is whether to encode functionality as net structure or as inscriptions. Try making a version of your Turing machine with as few places/transitions as possible and another version using as simple inscriptions as possible. Compare their output using simulation or state space analysis. What are advantages and disadvantages of each of the models?

Proposed task 1) Try running your Turing machine on a concrete Turing machine multiplying numbers in unary input formal and one multiplying numbers in binary format. How many steps are required to compute 5 * 3 and 10 * 6 on each of these? what does this say about the execution time of multiplication?

Proposed task 2) A non-deterministic Turing machine allows the state function to not only contain one result for each state-letter pair, but rather a set. A common problem in computer science is whether P = NP, i.e., whether deterministic and non-deterministic Turing machines has the same execution power in polynomial time. Implement a non-deterministic Turing machine (depending on your solution to Task 1, this is trivial) and run it on a non-deterministic Turing machine implementation of the subset sum problem (which is a simple NP complete problem: given a set of

integers, does any non-empty subset of the numbers sum to zero?). Investigate the state space of such an execution. You will notice that the state space grows extremely fast, so state space exploration is not a viable option for larger configurations. Try instead to modify your model so it generates log files using the ProM CPN library and import them into ProM for analysis.

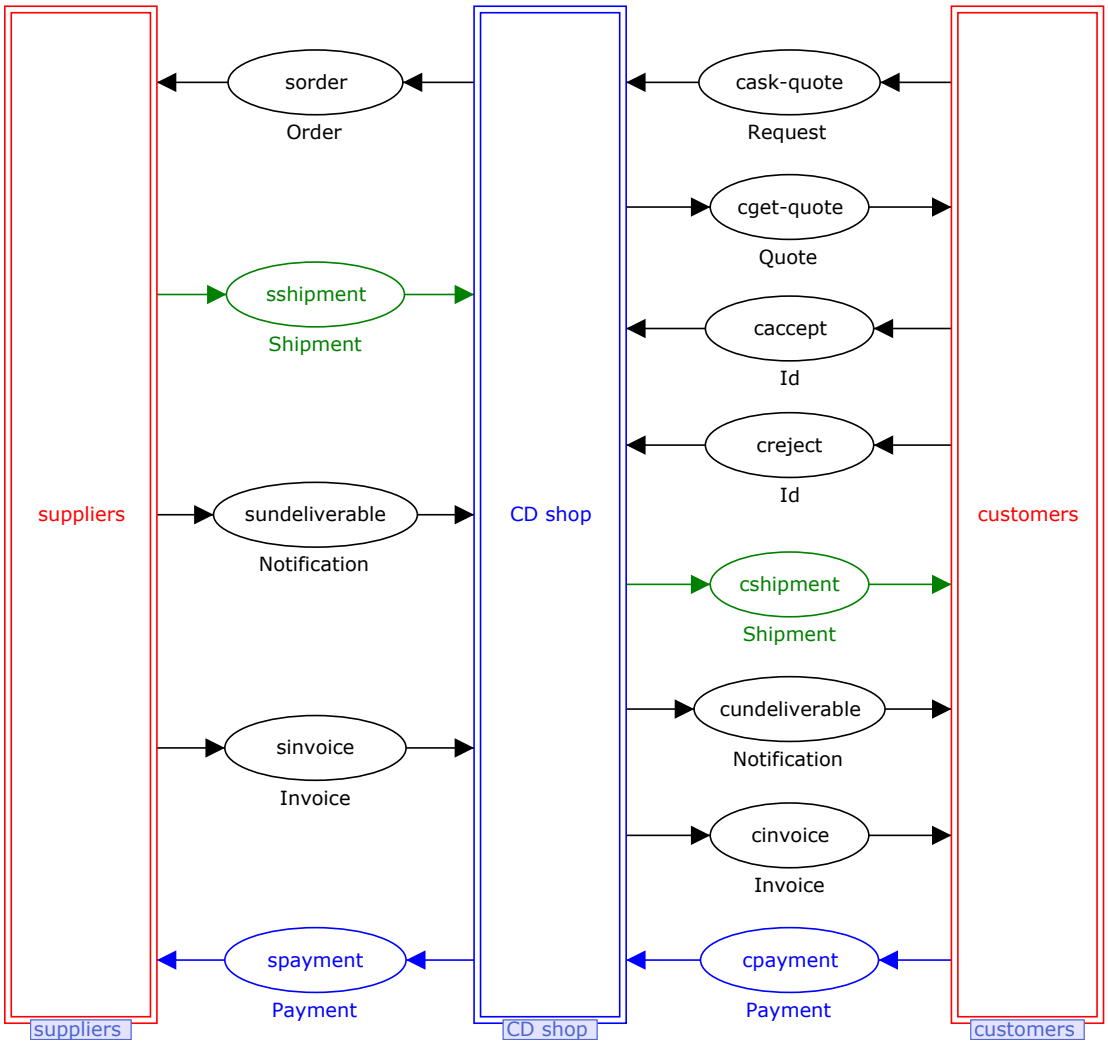
Proposed task 3) When explaining computer science and computability to people with little or no knowledge of computer science, formal definitions is not the way to go. A couple of my friends built this cool Turing machine in Lego: http://tinyurl.com/lego-turing. Use your Turing machine CPN model and the BRITNeY Suite to build a easily understandable graphical Turing machine.



Project 5: CD Shop

Modelling of processes from a detailed description and template.
Focus: Business process modelling and performance analysis.

The setting is a CD shop. The top page of a CPN model modelling a CD shop can look like this:



Customers can request quotes (i.e., offers) for CDs. The CD shop sells a limited collection of CDs and a customer may ask for CDs that cannot be offered because they are not the CD shop's collection. Moreover, a customer may ask for a single CD or multiple CDs (in fact even the same CD may be ordered multiple times). The CD shop responds to every request with a quote (i.e., an offer) indicating the CDs it can offer and the total price. The customer can then accept or reject. If the customer rejects, no further actions are needed. If the customer accepts the quote, then the CD shop tries to obtain the CDs mentioned in the quote from it suppliers. These are then shipped to the customer. If a requested CD is temporarily not available, the customer is notified and no attempts are made to deliver the CD later. After notifying the customer about undeliverable CDs and/or shipping the CDs available, an invoice is sent to the

customer. This invoice is based on the CDs actually delivered, i.e., the price is recalculated if some CDs mentioned in the quote are unavailable. Finally, the customer pays the invoice.

In turn, the CD shop orders CDs from specific suppliers, i.e., for each CD there is a dedicated supplier. The supplier ships the ordered goods. However, sometimes part of the order may be undeliverable, and the CD shop is notified. After notifying the CD shop about undeliverable CDs and/or shipping the CDs available, the supplier sends an invoice to the CD shop. Finally, the CD shop pays the supplier.

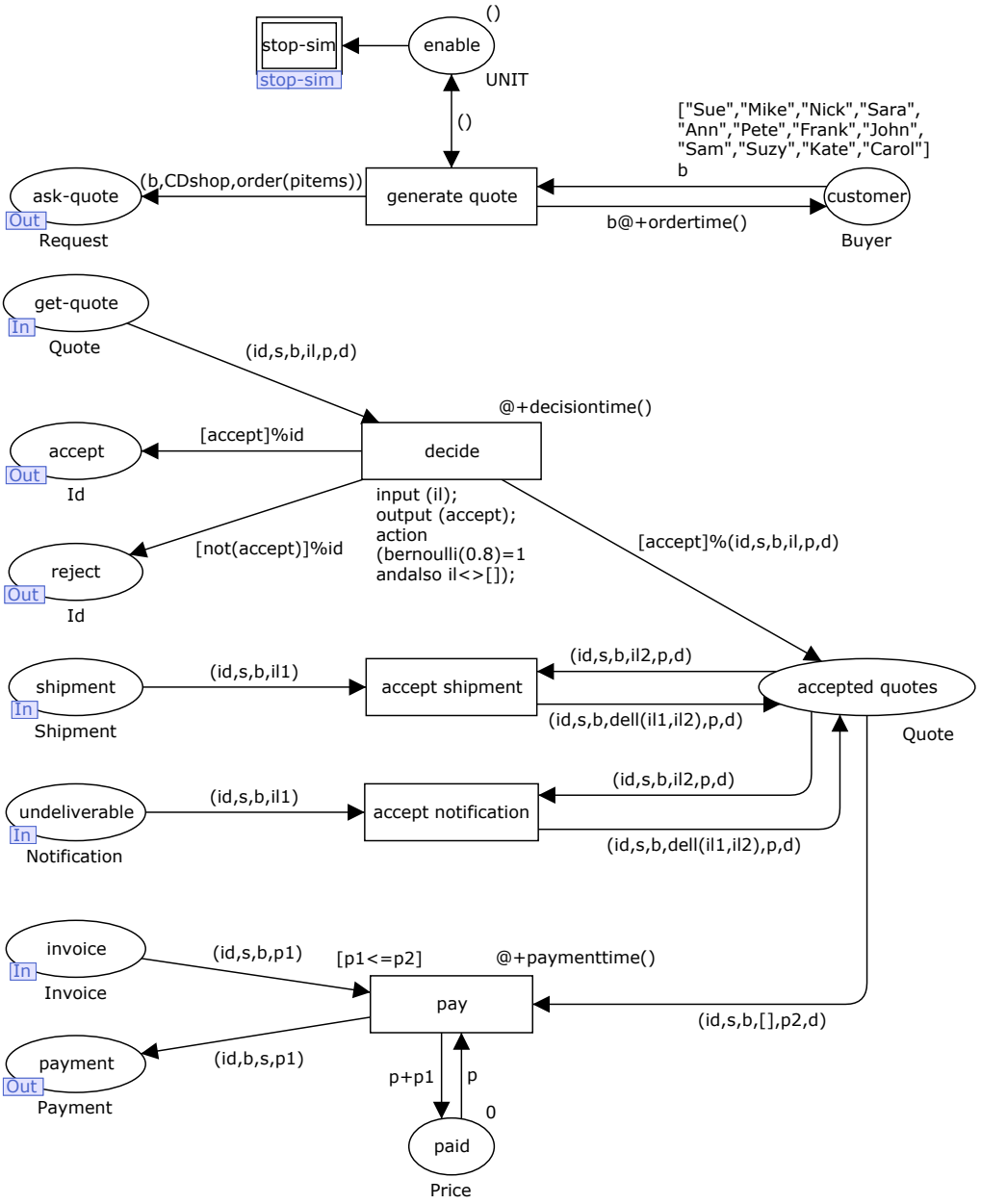
Shipments cost money. These costs are independent of the size of the shipment (1 CD or more).

Two aspects are important:

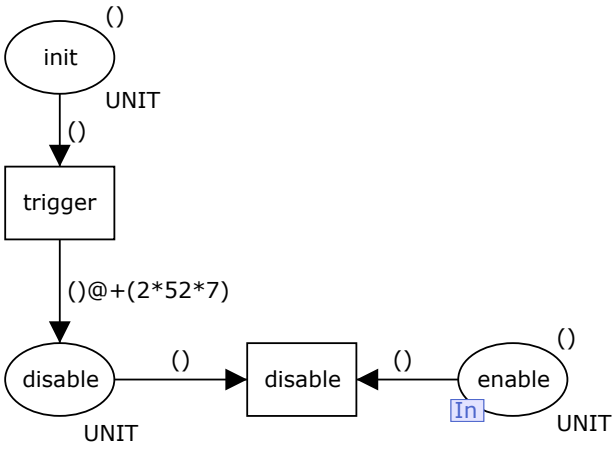
- The CD shop tries to make a profit by selling CDs for more than the costs of buying CDs from the supplier and the shipping costs.
- The customer is not only interested in the price, but also the time it takes to get the requested CDs.

When comparing different alternatives, we will focus on the above to aspects (profit versus response time).

Page *customers* (shown on the next page) models the behavior of (in this case 12) customers. Customers place requests by using the expression *order(pitems)*. Note that the list *pitems* also contains CD titles that do not appear in value *table*. When these are requested, the CD shop should mention these in the corresponding quote. Also note that the list of CDs requested is generated randomly, i.e., an arbitrary number of CDs may be ordered and the same CD may appear multiple times (meaning that the customer wants to have multiple copies). Every request should result in a quote received via port *get-quote*. Note that a quote is empty if no CDs are ordered that can be delivered by the CD shop. Empty quotes are always declined. Non-empty quotes are accepted with 80% probability. If a quote is accepted, the customer waits for the CDs to arrive. Some of the CDs mentioned in the quote may be temporarily non-available. In such cases the customer is notified (port *undeliverable*). Note that in principle one quote may result in multiple Shipment and Notification messages. Only when the customer has clarity about all items ordered, payment can follow. Payment is triggered by an incoming invoice. Note that the price of the invoice may be lower that the price mentioned in the original quote because of the unavailability of items. Note that monitor *flowtime* connected to transition *pay* collects statistics about the time between issuing a quote and initiating payment (see expression *Mtime()-d* in monitor definition). Here it is assumed that the proper day *d* is mentioned in the quote sent by the CD shop! On average the customers generate a new request every 4 days (Poisson arrival process) and take two days to decide on a quote (negative exponential distribution). Payments take seven days on average (also negative exponential distribution).



Page *stop-sim* is used to determine the duration of a single simulation run. In the initial setup a simulation period of 2 years (2*52*7 days) is used. Using "CPN'Replications.nreplications 10" ten simulation experiments are performed. On a modern laptop this takes about 10 seconds (depends of course on the exact model). Increase the length of the simulation period or the number of replications if the confidence intervals are too wide to make proper conclusions. Note that one can ignore startup effects.



CPN'Replications.nreplications 10

Task 1) Get the model fragment from http://tinyurl.com/cdshop-model. Get acquainted with the operation of the customer page and the supplier page (not shown here). Make sure you understand the declarations.

Task 2) Implement the CD shop page. The CD shop should reply to each request with a quote indicating what CDs can (potentially) be offered (only items from table may be offered) and the overall price (derived from table).

For accepted quotes, orders are placed with the corresponding suppliers (one order per item). Note that a quote for 5 CDs hence results in 5 orders. Moreover, each order has a unique id (issued by the CD shop) and is associated with a particular quote. Also note that suppliers are unreliable, i.e., 10% of the items ordered are not delivered. If an item is not delivered by the supplier, then it is also not delivered to the customer by the CD shop.

The CD shop waits until all orders related to a quote have been processed, i.e., there are no partial shipments. Everything that can be shipped is grouped into one shipment. Similarly, only one notification is sent if some of the items cannot be delivered. Note that there should be no empty shipments, i.e., if none of the requested CDs can be shipped, then obviously no shipment is needed.

The CD shop sends an invoice based on the items actually shipped. If some items could not be delivered because the supplier did not ship them, then the price needs to be recalculated using *table*.

The CD shop pays invoices send by suppliers for ordered CDs and handles payments made by customers.

Every shipment costs 2 euro. This is independent of the number of items and holds for both shipments from a supplier to the CD shop and from the CD shop to a customer.

The CD shop calculates the cash flow at the beginning of every week, i.e., money received minus money paid. Note that this so-called week profit is the sum of payments made by customers minus the sum of payments made to suppliers minus the costs for shipping goods (2 euro per shipment). This profit per week should be reported by the subprocess CD shop. Moreover, the total profit over the entire simulation period should be calculated (sum of week results).

Use simulation-based performance analysis to find estimates for the flow-times and profit.

Task 3: Supplier Reliability) Take the model made for Task 2. Investigate how sensitive the flow times and week profits are for various settings of supplier reliability. In the initial setting 90% of the ordered items are delivered (90% reliability). Provide the 90% confidence intervals for flow times and week profits assuming 100% reliability, 90% reliability, 80% reliability, and 50% reliability. Note that (unlike all other tasks) only the supplier page needs to be changed.

Use simulation-based performance analysis to find estimates for the flow-times and profit.

Task 4: Ship After 14 Days) Take the model made for Task 2. In this base model, CDs are shipped to the customer in a single shipment per accepted quote.

Modify the base model such that partial shipments are made after 14 days:
If all items for a quote have been delivered (or known to be undeliverable) within the first 14 days, then ship the items to the customer as soon as possible (e.g., after four days).

- After 14 days (counting from acceptance of the quote), ship the goods that have been delivered so far. For example, if two of the three CDs have been delivered by the suppliers after a 14 day period, then ship the two CDs and subsequently wait for the third one to be delivered (or until it is known that it will not be delivered due to supplier unreliability).
- If after another 14 days more CDs have been delivered but not all, ship the CDs received in the second 14 day period. Repeat this for the next 14 days, etc. Note that for a quote involving *k* items there may be between *1, 2, … k-1* or *k* shipments.
- At any moment; if for every item mentioned in the quote there is clarity (either delivered by supplier or undeliverable), then ship the remaining items.
- There cannot be any empty shipments to the customer.
- Send an invoice after all items have been shipped.

Use simulation-based performance analysis to find estimates for the flow-times and profit.

Task 5: Order Only Every 14 Days) Take the model made for Task 1. In this base model, orders to suppliers are always for a single item thus resulting in considerable shipment costs (2 euro per CD).

Modify the base model such that only periodically orders are placed in batches. Every 14 days at most one order is sent to a supplier. For example, all items of accepted quotes that can be supplied by "bol.de" are consolidated in a single order and this is repeated every 14 days. An order may involve multiple items related to multiple quotes.

If no items are needed from a supplier because there was no demand in the last 14 days, then no order is placed and only after another 14 days an order is possible.

Items ordered from suppliers are not linked to a particular quote (unlike Task 1). Hence, items shipped from the supplier to the CD shop need to be distributed over pending quotes. A received item may be allocated to any quote requiring the item.

Due to unreliability of suppliers, on average 10% are not delivered. These undeliverable items are randomly associated to pending quotes, i.e., like the items shipped from the supplier to the CD shop, also notifications need to be distributed over quotes.

Use simulation-based performance analysis to find estimates for the flow-times and profit.

Task 6: Re-order Until Shipped) Take the model of Task 5. However, now the CD shop continues to order items until they are actually delivered by the corresponding supplier. If the CD shop is notified by the supplier that an item cannot be delivered, then the CD shop simply reorders it for the next period. This is repeated until it is shipped from the supplier to the CD shop. Therefore, the CD shop is able to ship all items associated to quotes (eventually). Despite unreliable suppliers (10% cannot be delivered), the CD shop is 100% reliable for its customers.

Proposed task 1: Five Copies in Stock) Take the model made for Task 2. However, now the CD shop keeps a small inventory CDs.

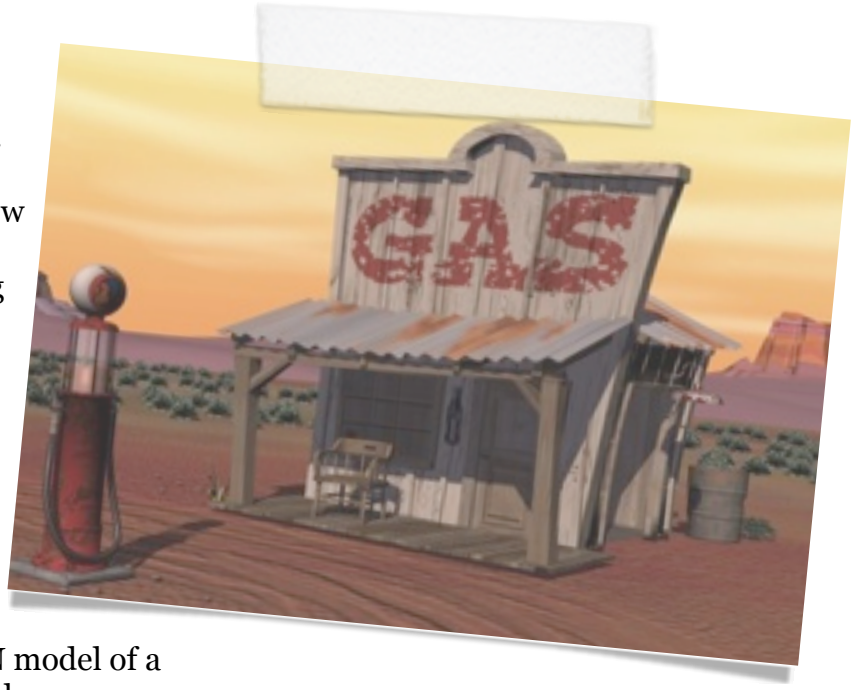
Modify the base model such that:

- For every CD mentioned in *table*, check the physical stock every week (every 7 days).
- If less than 5 copies of a CD are in stock (and not allocated to a particular quote), then order enough copies to increase the stock to the target level of 5. For example, if the stock of a CD is 3 at the end of the week, then 5-3=2 copies are ordered.
- An order may consist of multiple copies of the same CD. However, there is no need to combine different CDs in one order.
- Due to unreliability not all items need to be delivered. However, since the stock is checked each week, the missing items are re-ordered automatically.
- Items received from suppliers need to be distributed over pending quotes.
- The CD shop is reliable, i.e., all items in a quote are shipped eventually.

Project 6: Gas Station

Modelling of processes from an abstract description only.
Focus: Business process modelling and analysis.

Consider a modern gas station with a number of customers, six different pumps and one operator. Construct a CPN model that captures the workflow of customers arriving at the gas station for buying gasoline, including selecting the gasoline, pumping the desired amount of gasoline, and paying. Payment could be by credit card or cash, and there is also the option of prepayment.



Task 1) Construct a CPN model of a gas station as described above.

Task 2) If you have not already done so, make your gas station model timed, i.e., use transition delays to model that actions take time. How many customers can the station handle?

Proposed task 1) Suppose a competitor shows up next door, but the number of customers stay the same. What happens to the utilisation of the gas station?

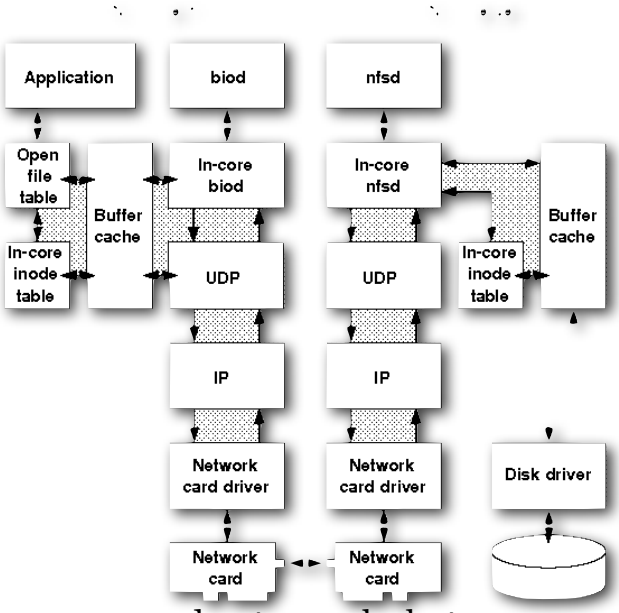
Proposed task 2) The gas station is part of a national chain of stations. Modify your model to reflect this. Consider whether you assume that the gas stations are the same (same number of pumps, etc.) or they can differ.

Proposed task 3) Export logs from your model and analyse them in ProM.

Proposed task 4) Make a nice visualisation of your gas station. You decide on the target audience for the visualisation.

Project 7: Distributed File System

Modelling and analysis of distributed protocols.
Focus: Modelling and performance analysis.



Distributed file systems play an important role in many modern systems. One of the better known distributed file systems is NFS (Network File System). Distributed files systems often need to balance simplicity of implementation and speed in use. NFS is a state-less file system, but does employ a caching machanism for improved speed. In this project we look at NFS and its performance.

Task 1) Get aquainted with NFS (v2 is described in RFC 1094; can be obtained at http://www.ietf.org/rfc/rfc1094.txt). If you are more adventurous, look at v3 (http://www.ietf.org/rfc/rfc1812.txt) or v4 (http://www.ietf.org/rfc/rfc3530.txt).

Task 2) Make a CPN model of NFS. The cache functionality should have special focus, and it should be possible to switch off caching.

Proposed task 1) Use state space analysis to check the caching mechanism in NFS. State some correctness criteria and analyse them.

Proposed task 2) Make sure your model is timed and use simulation-based performance analysis to compare the performance of NFS with and without cache. You can try comparing newer and older versions of NFS as well.

CPN Tools GUI

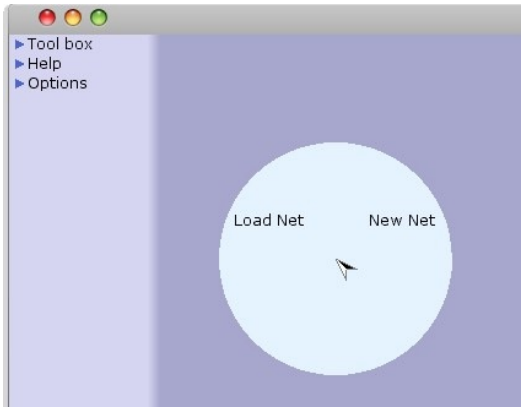
The Interface

When you open CPN Tools, the first thing you see is a window, containing an index on the left side and a workspace on the right.

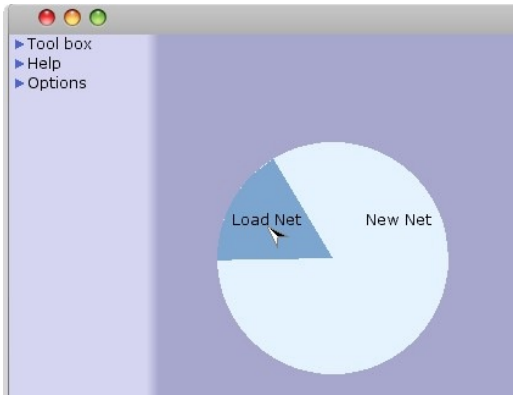
To get started working on a CP-net, you can either load an existing net or create a new one.

Loading and Creating Nets

To load an existing net, move the cursor to the workspace or to an empty part of the index. Press with the right button of the mouse and keep the button pressed. A circular menu appears - this is the workspace marking menu.

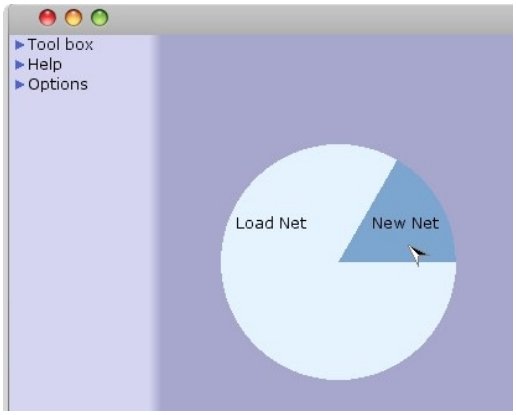


Keep the button pressed and move the cursor to the Load Net entry in the menu. The entries will highlight when the cursor moves over them - release the button when the Load Net entry is highlighted.



A file dialog appears from which you can select the net you wish to load. When you have selected a net and clicked OK, the dialog disappears, and the net appears in the index at the left side in the CPN Tools window. After a net has been loaded, the name of the net appears as an entry in the index.

Alternatively, you can create a new net by bringing up the workspace marking menu and selecting the New Net entry.



Remember to keep the right button pressed as you move the cursor around in the marking menus. If you release the button when no entry is highlighted, the menu disappears and no command is invoked. Press down the right button to bring the menu up again.

The Index

The index is located in the left side of the CPN Tools window:

The index contains:

- **Tool box**: A list of all the tools available in CPN Tools. See the Tool box entry below for more information.
- **Help**: Links to Help, Homepage, Report Bug and other webpages. See the Help entry below for more information.
- **Options**: Various options for, e.g. simulation
- The nets that are currently open in the tool

A blue triangle to the left of an entry in the index indicates that the entry can be opened to reveal more details about the entry. Click the triangle to open and close an index entry.

Overview of a Net

The index contains an entry for each net that is opened in the tool. If the net was loaded from a file, then the net entry in the index is labelled with the name of the net. In the figure, the net named "HierarchicalProtocol.cpn" has been loaded. If a new net was created, the net entry is labelled "New net.cpn".

To start working on a page in the net, the net entry in the index must be opened. The entry may already be opened. Click on the triangle next to the name to open the entry, if it is not already opened.

A net entry contains:

- **History**: the list of commands that have been performed on the net
- **Step**: the number of steps that have executed in a simulation
- **Time**: the current model time
- **Declarations**: the declarations of colour sets, functions, constant values, etc. All declarations are written in the CPN ML language.
- **Monitors**: all monitors defined for the net.
- Page entries for pages in the net

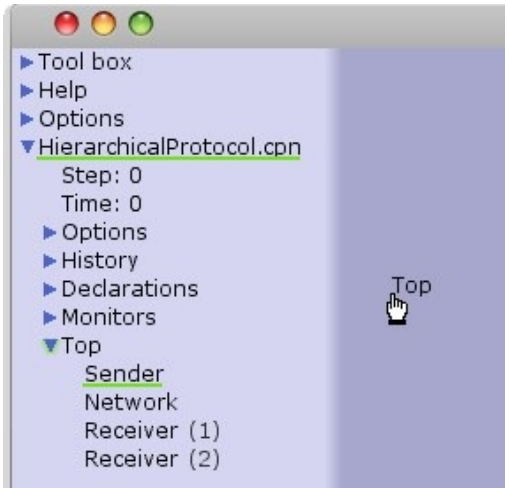
All pages in a net are accessible through the index. The hierarchical structure of a net is reflected in the index. The entry for a subpage appears under the entry for its superpage. Subpage entries are visible in the index when the corresponding superpage entry has been opened.

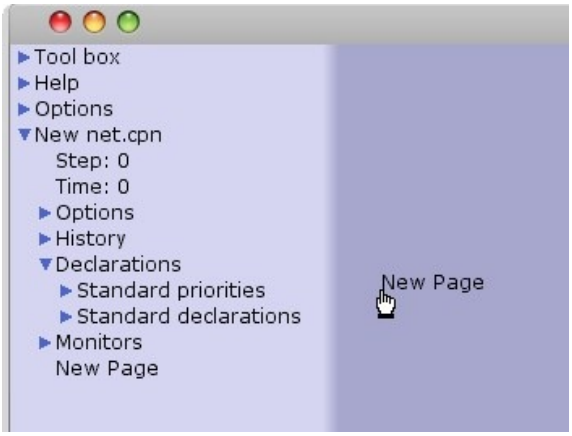
Opening the entry for the "Top" page shows that it has four subpages: one instance of the "Sender" and "Network" pages, and two instances of the "Receiver" page.

If you have created a new net instead of loading one, there will be only one page, called New Page. This page is created automatically with the new net, and is empty so you can start creating net objects on it.

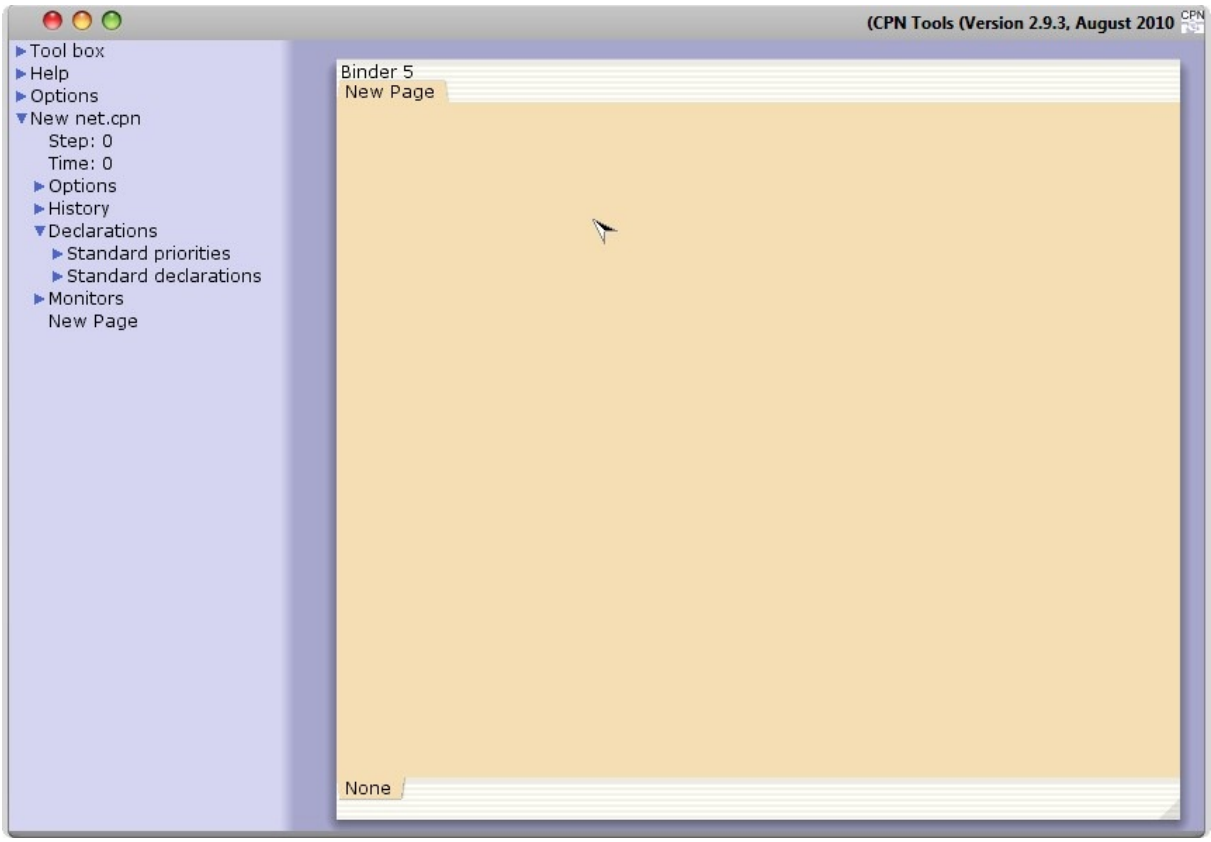
Opening a Page

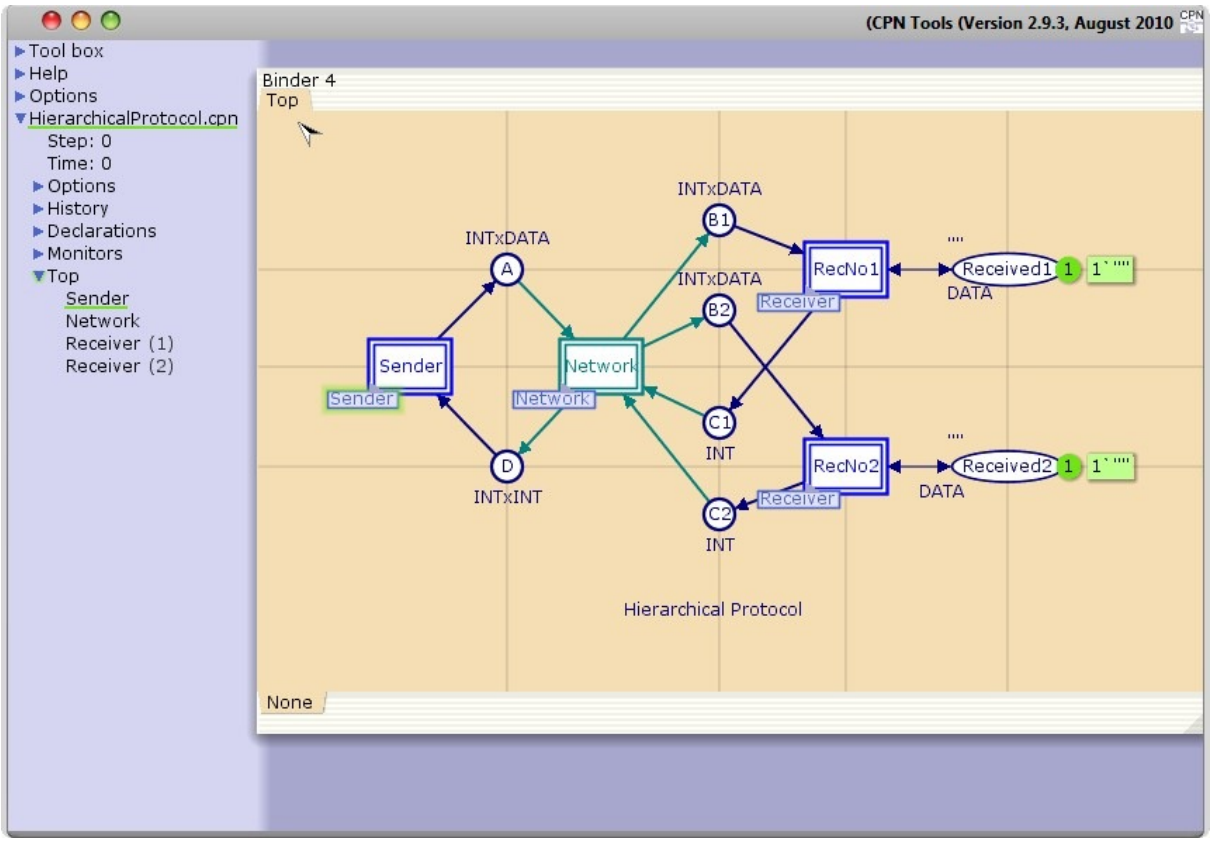
To open a page in the net, move the cursor over the name of the page, press down the left mouse button and drag the page name to the workspace. When opening or creating a net, one or more of its pages may already be opened. Here the page named "Top" is being dragged to the workspace. If you have a new net, drag the page "New Page" to the workspace to start creating a net.





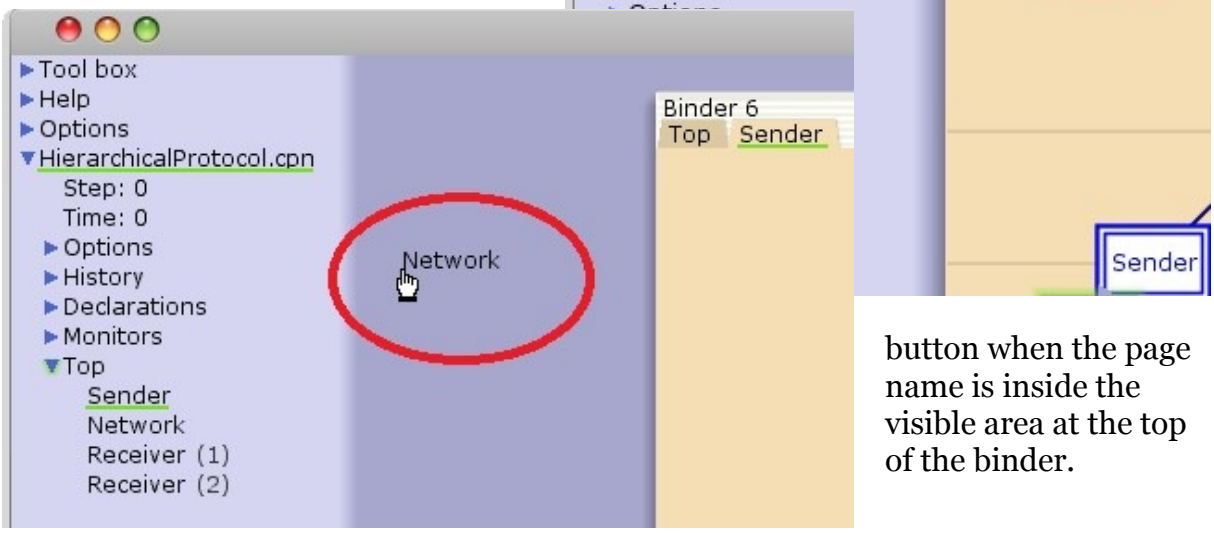
Release the mouse button. The page now appears on the workspace in a binder. If you have dragged the empty page out, you probably want to start creating objects etc. See the Edit the net page for more information.



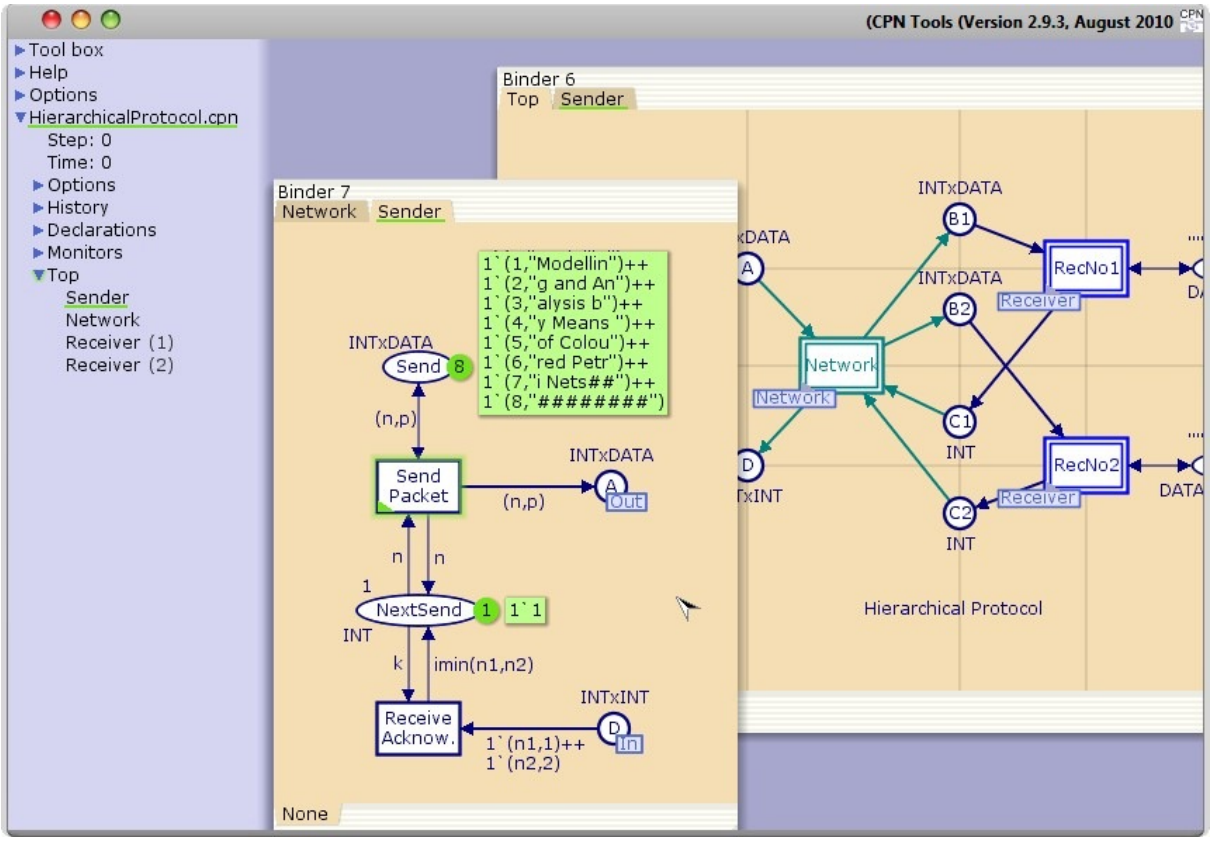


Multiple Pages and Binders

The pages can be dragged from the index to the workspace and dropped in one or more binders. To drop a page in a binder, release the mouse

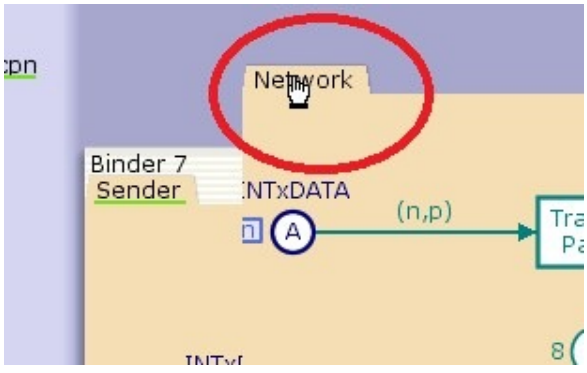


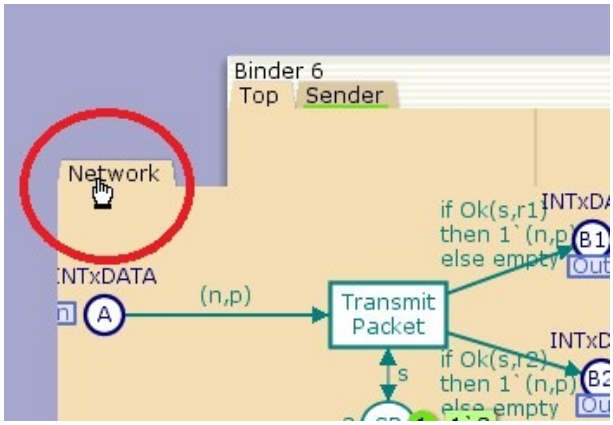
button when the page name is inside the visible area at the top of the binder.



To drop it in a new binder, drop the name outside of the binder area.

You can place several pages in the same binder, and you can have the same page in more than one binder. A binder can only contain pages from one net.
To move a page from one binder to another, press with the left mouse button on the page tab at the top left corner of the page and drag the page to an existing binder. You can also put it in a new binder by releasing the page somewhere on the workspace in the same way as when dragging from the index.





The position of the binders are saved when you save a model.

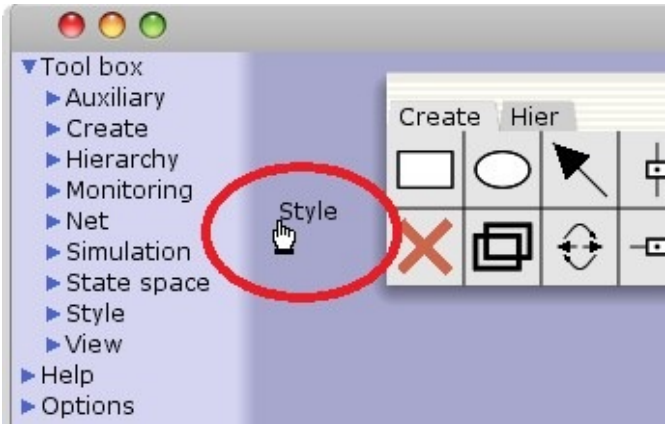
Tool box

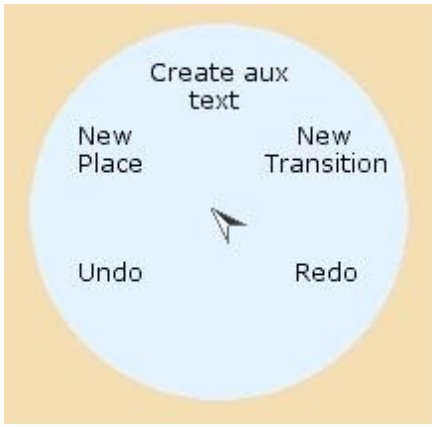
Tools for creating new CP-net elements, simulating nets, changing line colors and line widths, etc., are accessible from tool palettes in the tool box in the index and through marking menus on the various objects in the workspace.

To access a tool palette in the index, open the Tool Box entry in the index.

Drag a palette to the workspace the same way as you drag a page: Move the cursor over the name, press the left button, drag the name to the workspace, and release. Like pages, tool palettes can be placed in binders together or separately.

To access a tool in a marking menu, press down the right mouse button over an object and choose the command you wish to perform. The marking menus are context-sensitive, i.e. their contents change depending on where you bring them up. If you, e.g., press down the right mouse button in an empty area on a page, a menu will appear with commands for creating new CP-net elements: places and transitions.





Help

To open a help page from the index drag it to the workspace, like the interaction for opening a page (see the Opening a page entry above).

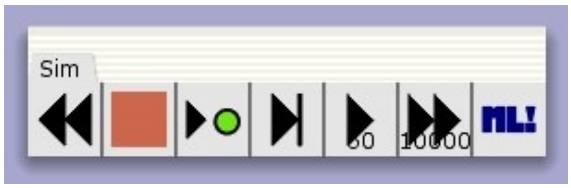
You can also access the help pages for CPN Tools at http://wiki.daimi.au.dk/cpntools-help

Simulation in CPN Tools

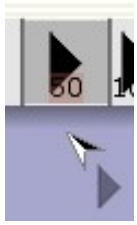
Run a Simulation

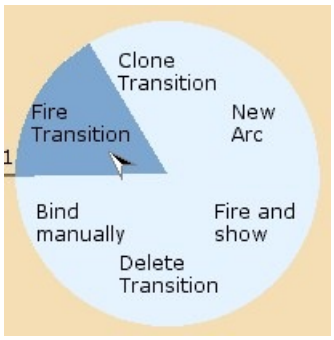
When you have loaded or created a CP-net, and it is successfully syntax checked (i.e. there are no orange or red auras), you can start running simulations.

A simulation report will be saved if the appropriate options have been checked.



To start a simulation, drag the simulation tools from the index to the workspace. The palette contains 7 tools, we will only worry about the first 6: rewind, stop, manual simulation, single step, play, and fast forward. Play runs a simulation with intermediate feedback. This can be stopped using the stop tool. Single step allows you to execute a single transition at a time. Manual simulation furthermore allows you to select the binding to execute. Fast forward allows you to execute a lot of transitions very fast, only showing feedback when the entire simulation is done. After simulation, rewind brings the model back to the initial state.





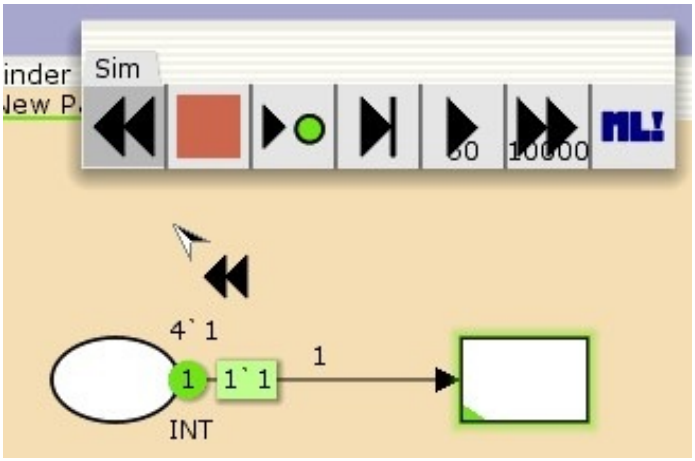
Now select and apply one of the simulation tools from the tool palette. Applying, e.g., the Play tool from the simulation palette will cause the simulation to run a number of steps as specified in the tool cell.

It is also possible to execute single simulation steps if you manually choose bindings or apply a simulation tool from the transition marking menu to an enabled transition.

Initial State

Use the Rewind tool from the Simulation Tools to return a CP-net to its initial state (initial marking).

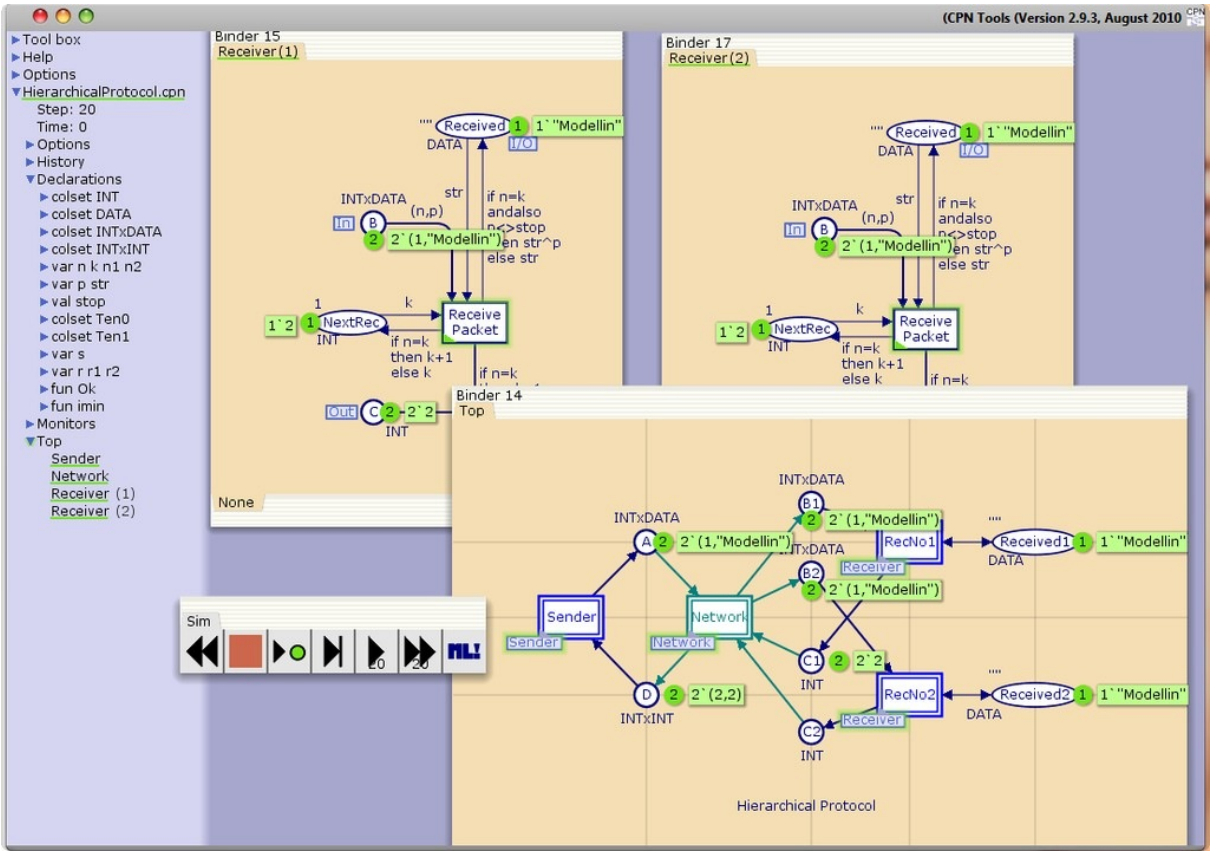
If you have changed or added initial markings on some of the places, you may have to use the Rewind tool to revert the net to the initial state.



Simulation Feedback

While a simulation runs, the following simulation feedback is shown:

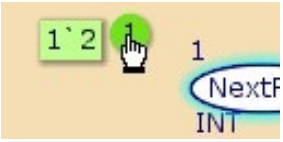
- Current markings of places are shown near the places.
  - The number of tokens on a place is shown in a green circle.
  - The corresponding token values are shown in green boxes
- Green auras around enabled transitions and green underlines on the pages with enabled transitions (shown in the index and on page tabs)
- Steps and time shown in the index under the net being simulated



A green status bubble will also appear when a simulation stops. A speech bubble will show the reason why the simulation stopped.

Moving and hiding current marking information

Sometimes the simulation feedback covers parts of the net, making it hard to read. Therefore, it is possible to move the current marking information around by dragging it to a new position. Dragging the green circle will also move the green box with token values, but dragging the box will move the box independently from the circle.



The positions of current marking information are saved when you save a model.

The token values can also be hidden. Click on a green circle to hide/show the corresponding token values.

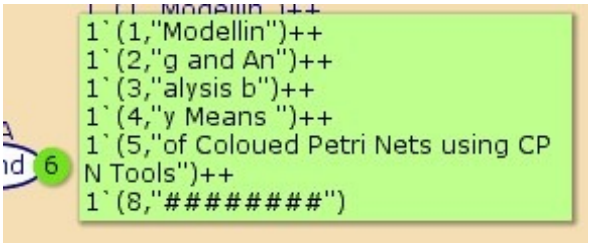
Fast-Forward Simulations

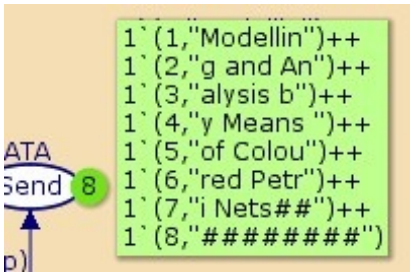
Simulation feedback is not updated during a fast-forward simulation, but it is updated after the simulation has completed.

After applying the fast forward tool, a light purple status bubble will indicate that a time-consuming operation, i.e., a long simulation, is currently being executed.

Size of current marking box

The green boxes with token values will show one token value per line when the token values are relatively short.

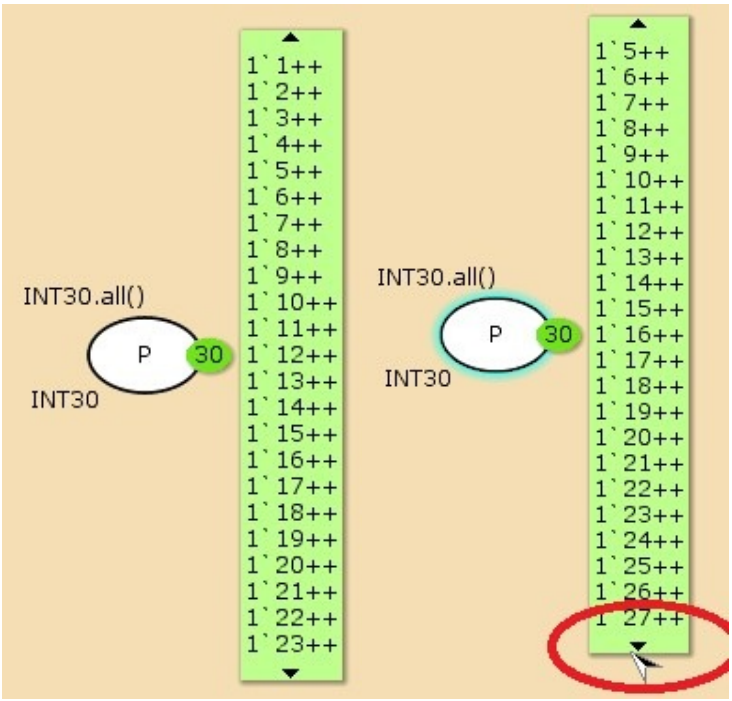


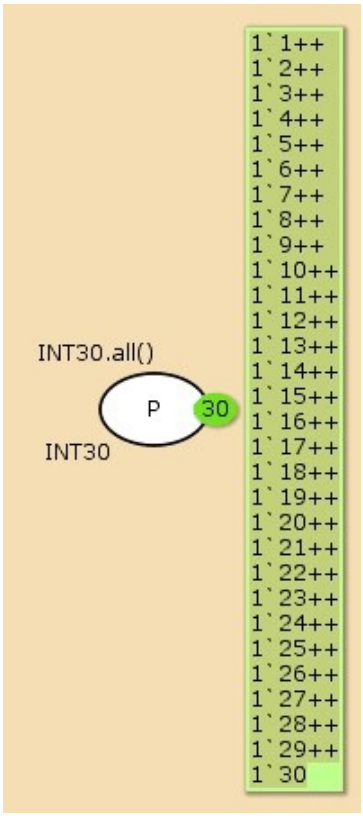


When the length of a token value is long, i.e. it exceeds a predefined width, the token value will be wrapped and shown on two or more lines.

If the green box with token values is more than ca. 20 lines long, then only ca. 20 lines will be shown at one time, and two black arrowheads for scrolling will be added to the top and bottom of the green box. Click on an arrowhead to scroll in the token value text.

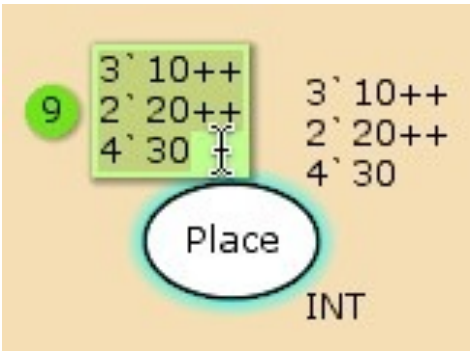
If the green box has arrowheads for scrolling, then all token values can be viewed by entering text-edit mode in the green box.





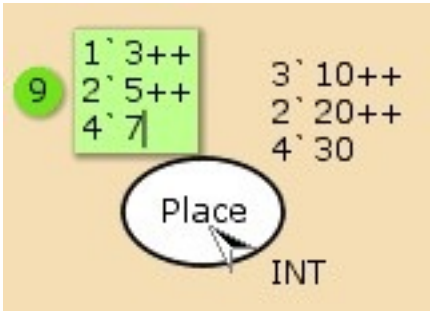
Change marking during simulation

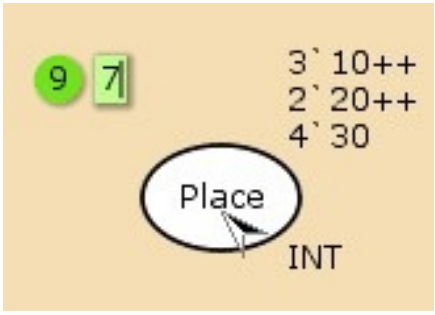
During simulation you can change the current marking of a place. Simply click on the marking as shown below to enter text edit mode, and type in the new inscription. When you exit text-edit mode, the place will have the new inscription.



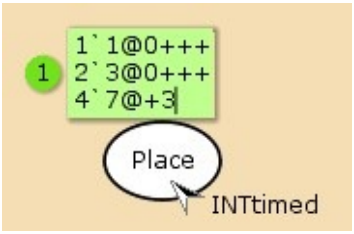
If the colour set of the place is not a timed colour set, then the new inscription must evaluate to either a multi-set of values from the colour set of the place, or a single value from the colour set of the place.

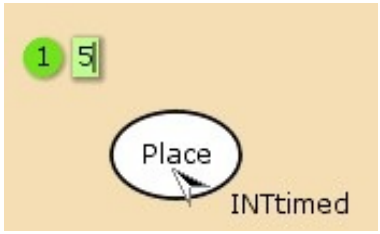
If the colour set of the place is a timed colour set, then the new inscription must evaluate to either a timed multi-set of values from the colour set of

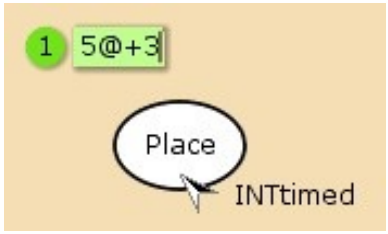




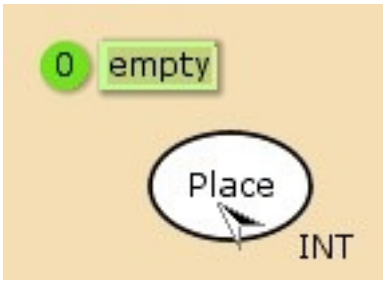
the place, a single untimed value from the colour set of the place, or a single timed value from the colour set of the place.

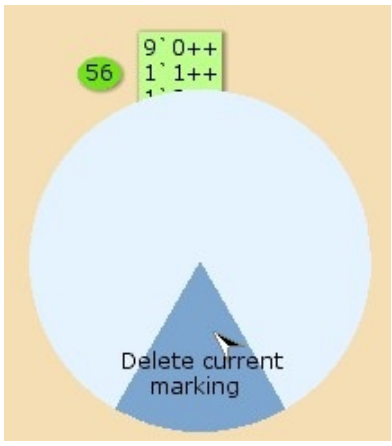




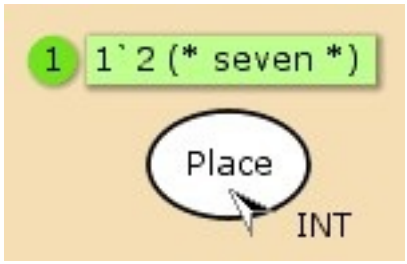


Use the Show Marking entry of the Place marking menu to update or create the current marking of a place instance. You can change the marking of a place to be empty by applying the Delete Element tool to the marking information. This tool can be found in the Current marking marking menu.





If you write an illegal current marking on a place. The old marking is inserted after a syntax check with the illegal marking afterwards in comments.

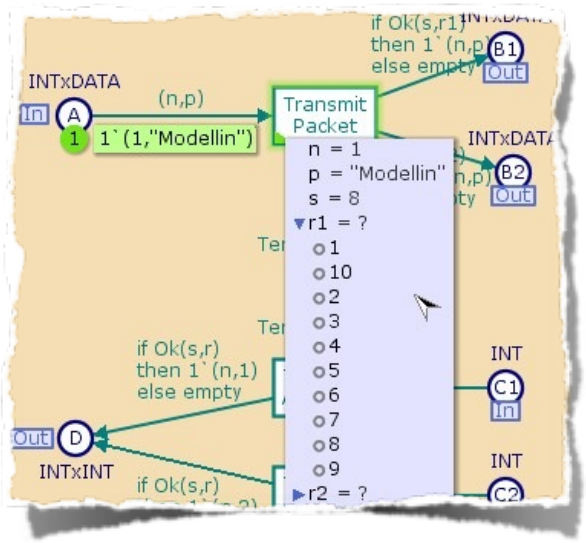


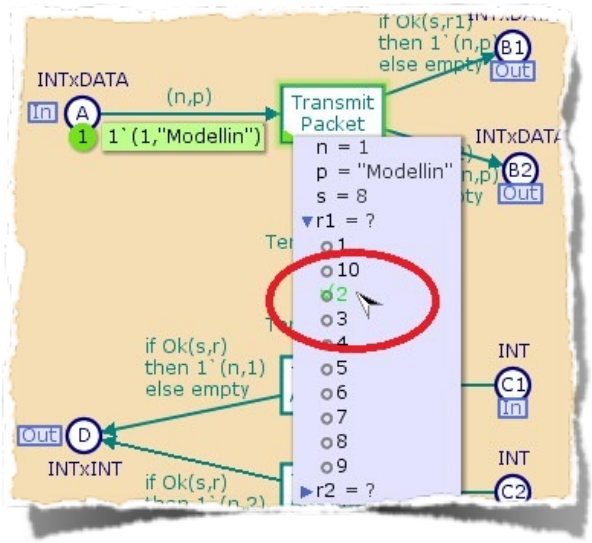
Manually Choose Bindings

It is possible to choose which binding should be used when firing a transition. This is done by using the bind manually tool or clicking on the green triangle in the lower left corner of an enabled transition. This will open a binding index showing the variables and the values they can be bound to. A "?" after a variable indicates that no value has been chosen for that variable.

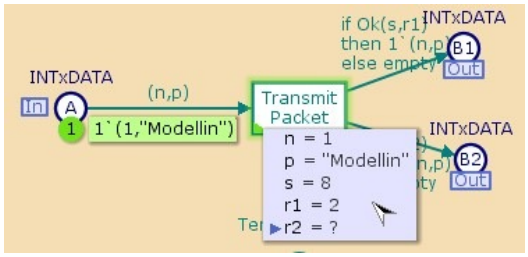
To choose a value for a variable simply click on the value. The binding index will then change, removing all the values that are no longer possible.

To fire the transition with this binding use the single step tool or, if all variables have been bound, just click on the binding index. It is possible to partially bind a transition. A value will then be chosen for the variables that are not bound when the



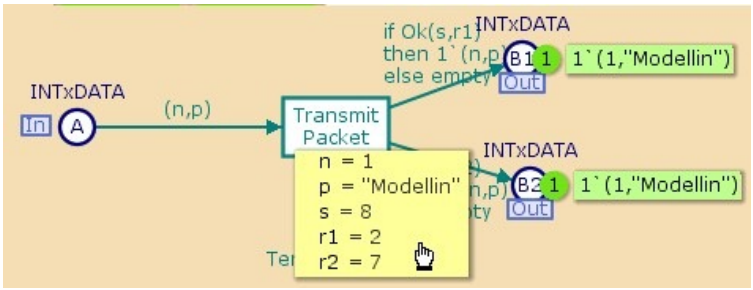


transition is fired. After firing the binding index changes background colour and shows the binding that was used.



To close the binding index after firing simply click on it. It is possible to abort manual binding using a marking menu on the binding index. Manual binding will also be aborted if any operation communicating with the simulator is performed.

It is possible to move the binding index to another position. The position will be saved when you save a model.

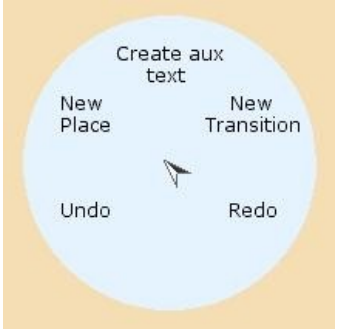


Editing in CPN Tools

Creating places, transitions and arcs

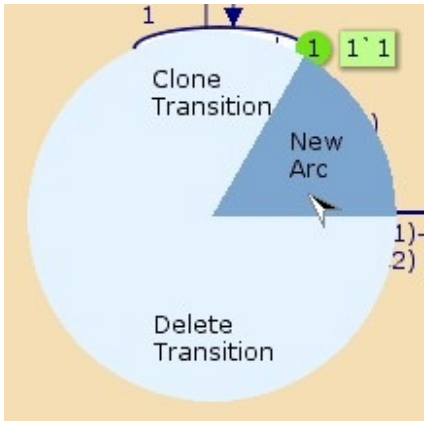
On the pages in a net, you can create places, transitions and arcs using either Marking menus or Palettes. Start by dragging a page out from the index.

Marking Menus



To create a place or a transition, bring up the Page marking menu by pressing and holding down the right mouse button on the page. Now select New Transition or New Place.

An arc must go from a place to a transition or from a transition to a place.



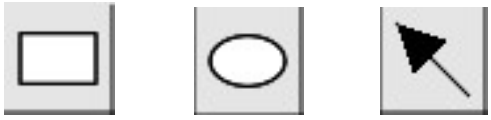
To create an arc, hold down the right mouse button on the place or the transition where you want the arc to begin. Either the Place marking menu or Transition marking menu appears.

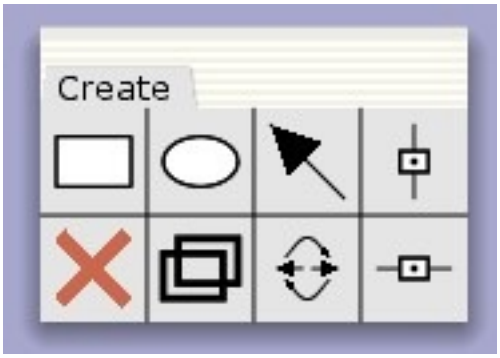
Select New Arc to start an arc. An arc will be attached to the cursor, and you can drag it to the transition or place where you want the arc to end. Click on the background to add bend points or on a transition or place to finish the arc. You can also add more bend points later by dragging the arc where you want the bend point to occur.

You can abort the creation of an arc by applying the Drop Tool tool before you attach the final endpoint.

Palettes

Use the Create tools Palette to create transitions, places and arcs. Select the desired element to create transition, place, or arc:



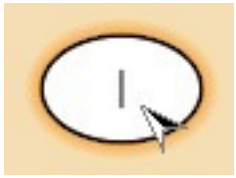


Click with the left mouse button where you want to create the element.

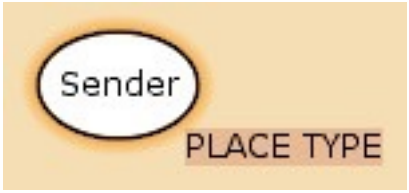
When creating an arc, you can create bend points by clicking on the page background. If you want to cancel the arc, click again on the palette cell where you picked up the arc tool. Alternatively, bring up the Workspace marking menu and select the Drop Tool entry or press Esc.

Adding Inscriptions

Places, transitions and arcs are provided with different inscriptions of different types, such as the color set or initial marking for places, the guard for transitions, and arc expressions for arcs.



Immediately after creating a place/transition/arc you are also in text edit mode, and can add the first inscription right away. In order to add inscriptions to places which are not in text-edit mode, click on the desired place. Now, the cursor is placed inside the place and you can edit the place name, which is the first inscription. When the cursor is inside the place, you can press TAB to go to the next inscription, which is the type/colour set of the place. You can progress further by pressing TAB again. The last thing you can add to a place is an optional initial marking for the place. Once inscriptions have been added, you can change them by doing this procedure again or, more simply, by clicking on the desired inscription.



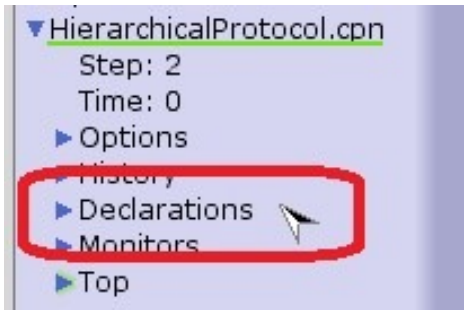
A similar procedure is available for transitions. First you specify the name of the transition. By pressing TAB, you can insert a guard. By pressing TAB subsequent times, you can do the same for the time delay, code segment, and for the priority.

For arcs, there is only one inscription available, which is the arc expression.

Note: When you have changed or added an initial marking, you might need to use the Rewind tool in the Simulation tools in order to see the token for the marking change/appear.

Declarations

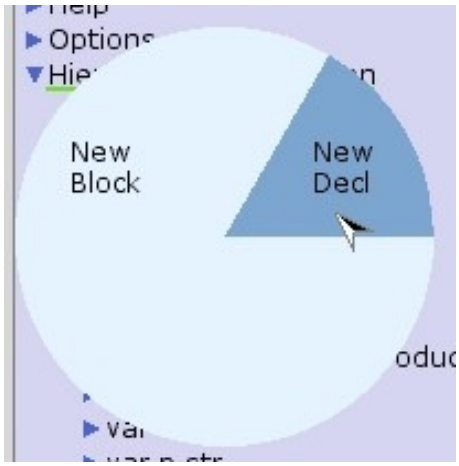
The declarations are located in the index under the Declarations entry under the net name.



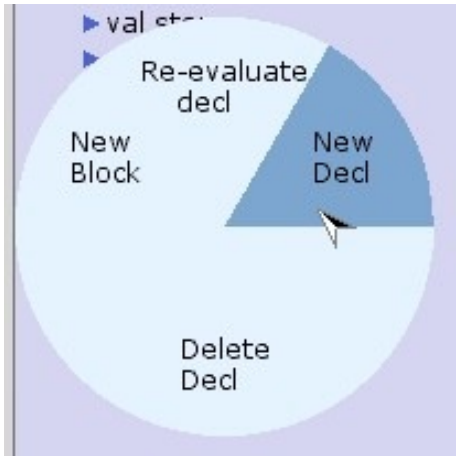
Clicking on the triangle next to the Declarations entry opens the declarations. Clicking on a triangle next to an individual declaration or clicking on the declaration text opens this declaration.

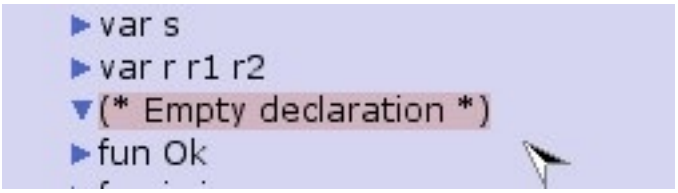
Add a declaration

To add a declaration, bring up the Declarations marking menu on the top declarations node or on one of the separate declarations in the index. Select the New Decl entry.

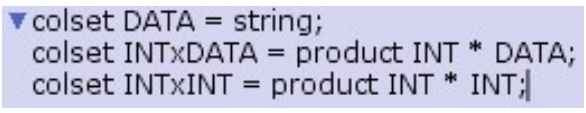


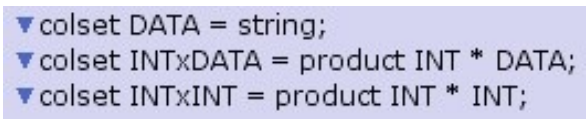
The added declaration will appear either at the bottom of the list of declarations or below the declaration where you added it. The declaration will be in text edit mode when it appears so it can be modified right away.





You can insert new lines by pressing Enter, and you can create a new declaration below the one you are editing by pressing Ctrl-Enter. You can also add a declaration to an existing declaration if you separate them with a ';'. When you finish text editing, the declarations will be split into separate declarations.

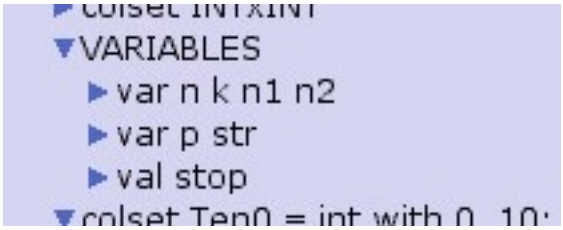


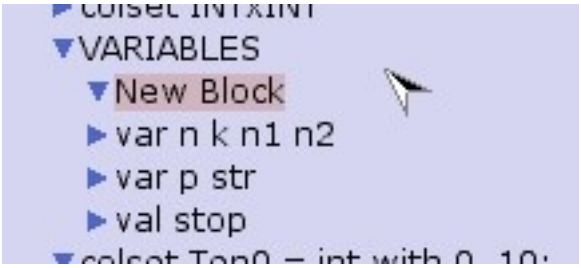


To finish text editing, click somewhere on the background or press the ESC key.

Add a declaration block

Declaration blocks are used to divide declarations into groups, often in larger nets with many declarations. Here, the variables are put together under a block called VARIABLES:



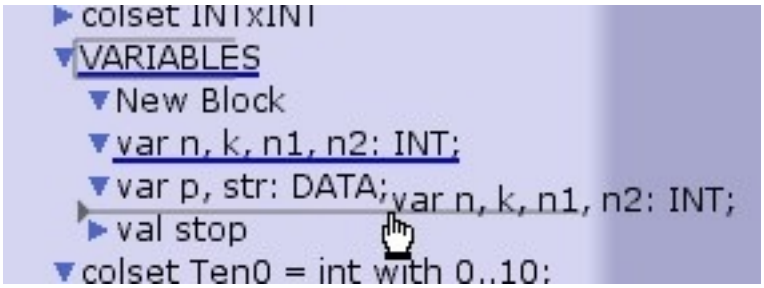


To add a declaration block, bring up the Declarations marking menu and select the New Block entry. The new block will appear at the bottom of the list or after the declaration where you added it, and it will be in text edit mode when it appears.

To add a declaration to a block, bring up the Declarations marking menu on the block and select the New Decl entry.

Moving a declaration or declaration block

It is possible to rearrange the declarations by dragging and dropping within the Declarations entry in the index. You can drag declarations into and out of declarations block to improve your overview of your declarations.

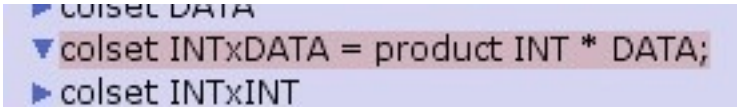


The only limitation is the syntax check, which requires declarations to come after those it depends on. Which in turn can be achieved by dragging declarations.

Adding declarations through keyboard shortcut

If you are editing a declaration or block (see below), you can add a new declaration by pressing Ctrl-Enter. If you are editing in the index, the new declaration will appear below the one you are editing. If you are editing in a text sheet, the new declaration will be opened in the same binder as the one you are working in.

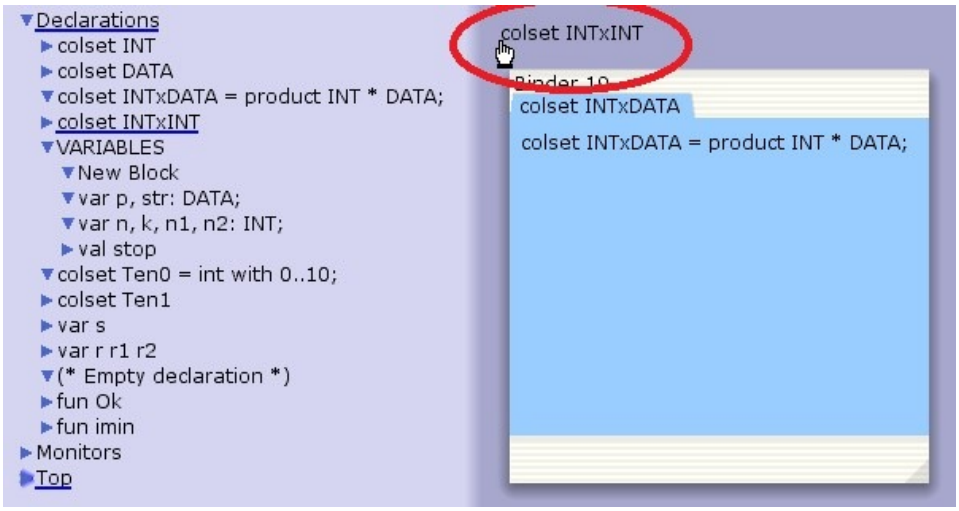
Edit a declaration



To edit a declaration or block, open it in the index by clicking on the triangle next to it, and then click on the declaration or block text. The redish color indicates that all text is selected.

To finish editing, click somewhere on the background or press the ESC key to exit text edit mode.

Alternatively, drag the declaration/block from the index to a text sheet.



Click on the text and edit it. To finish editing, click somewhere outside of the text or press the ESC key. The text will now be syntax checked, and any errors that might occur will be shown with highlighting. See Syntax checking for more information.

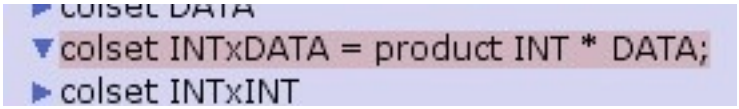
Editing the next declaration through keyboard shortcut

If you are editing a declaration or block, you can jump to the next declaration or block by pressing TAB. If you are editing in the index, you will jump to the next declaration or block in the index. If you are editing in a text sheet, you will jump to the next declaration in the same binder.

Edit Text

Text editing is used to change inscriptions, declarations, page names, net names, group names, options, tool options, etc.

Enter Text-Edit Mode



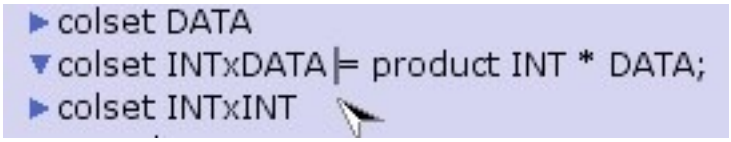
To enter text edit mode, simply click on the text. If you have a tool in the hand, use the long click. A redish colour indicates that all text is selected. Type to replace the selected text.

After clicking on an empty text, e.g. on a place with no name, the cursor will change to a blinking crossbar.

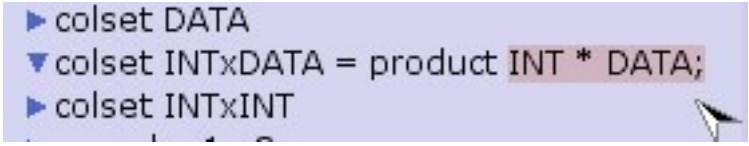
Leave Text-Edit Mode

To leave text-edit mode, click somewhere outside the text, or press the ESC key on the keyboard.

Editing Text



Text can be edited after entering text-edit mode. To insert new text within existing text, click on the text to position the cursor in a specific part of the text.



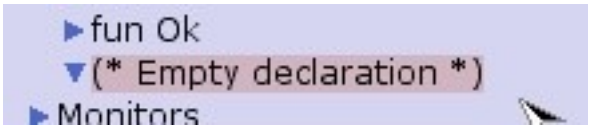
A subtext can be selected by pressing the left mouse button at one end of the subtext, dragging the cursor to the other end of the subtext, and then releasing the mouse button. Type to replace the selected text.

Cut/Copy/Paste

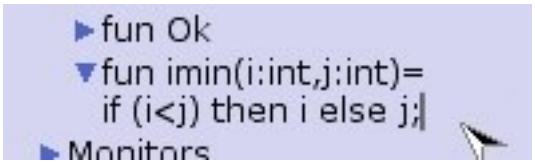
Text can be copied and pasted using keyboard shortcuts:

- *Ctrl-c* will **copy** the selected text.
- *Ctrl-x* will **cut** the selected text.
- *Ctrl-v* will **paste** the text that was most recently copied or cut.

Text can be copied, cut, and pasted within CPN Tools and between CPN Tools and other programs. For example, you can copy a text from another text editor (using any of the copy commands available in the text editor), enter text-edit mode in CPN Tools...



and paste the copied text using Ctrl-v.



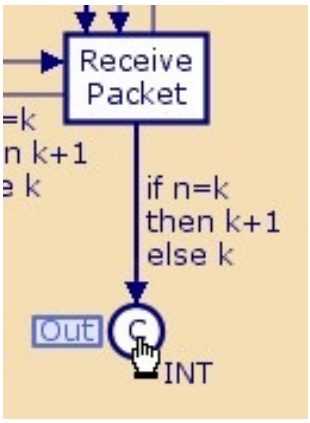
Navigating within a text

The cursor can be moved within a text by:
- Using the *left*, *right*, *up* and *down* arrow keys on the keyboard
- Using the *Home* and *End* keys on the keyboard
- Clicking with the mouse to choose a specific position

Editing the Layout

Snap to straight

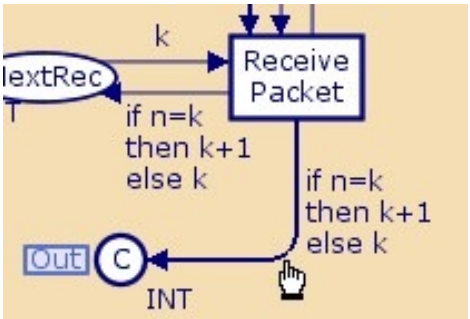
When you move places, transitions or bend points on arcs connected to other transitions or places, the moved object snaps to a straight line relative to the attached objects. This makes it easier to quickly do simple layout adjustments.

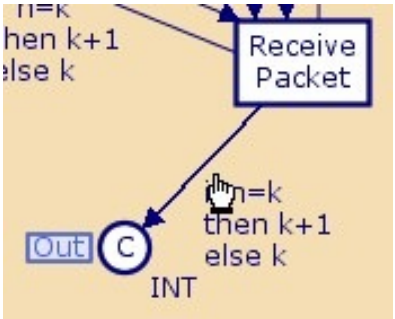


For example, when moving a place attached to a transition, the place snaps to vertical and horizontal straight lines relative to the transition.

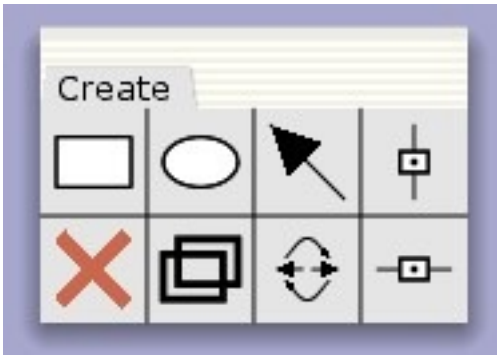
Bend points on arcs snap to straight angles relative to their connected nodes and other bend points.

If you move a bend point so it makes a straight line with the surrounding bend points, it snaps to the straight line and is automatically deleted when you release the button.



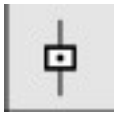


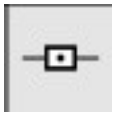
Magnetic guidelines



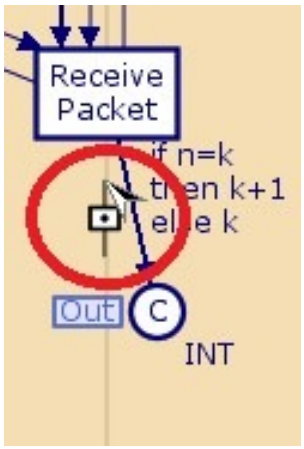
Magnetic Guidelines in the Create Tools make it easier to position objects in alignment.

You can add vertical and horizontal guidelines.

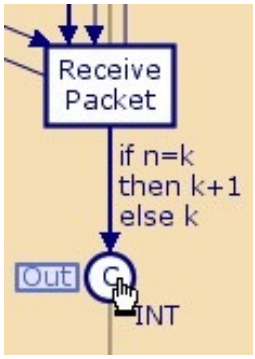




To add a guideline, click on the page background using either the vertical or the horizontal guideline-tool.

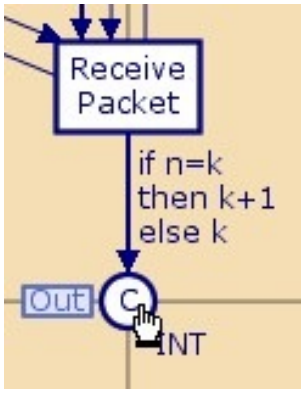


To attach places, transitions or bend points to the line, drag and drop them onto the line. The guidelines are magnetic, so when an object comes close enough it snaps to the line, and the line highlights to show that the object has snapped to it.



To remove an object from a guideline, simply move the object away.

Moving a guideline will result in moving all objects attached to the guideline. It will not snap objects when moving past them. To remove a guideline, delete it using the Delete element tool in the Create Tools.



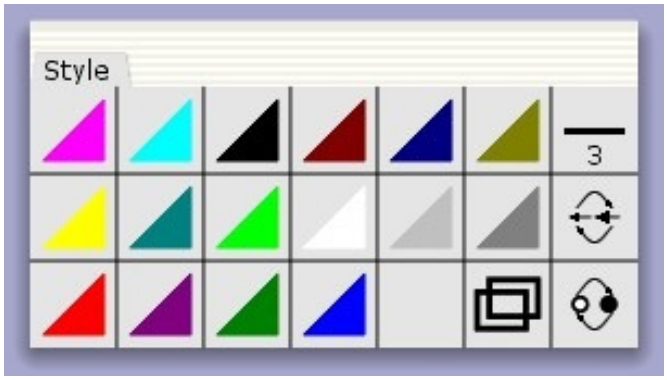
You can add an object to more than one guideline at the same time, if they are of different kinds (i.e. one horizontal and one vertical guideline), but only one of each kind (i.e. you cannot add an object to two horizontal guidelines).

Applying on of the Style tools to a guideline will apply the tool to all the elements attached to the guideline.

For more information about guidelines please visit the Magnetic guidelines page.

For examples of how to use guidelines you can look at the Distributed Database or Timed Protocol in the Example Nets.

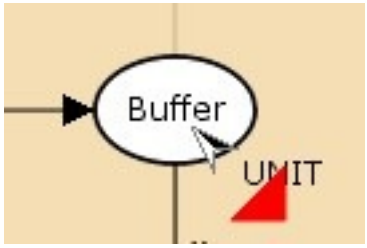
Edit Style Attributes

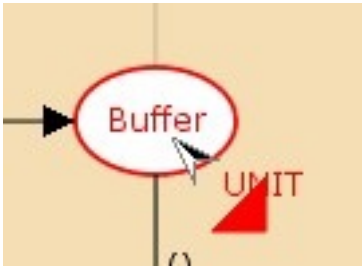


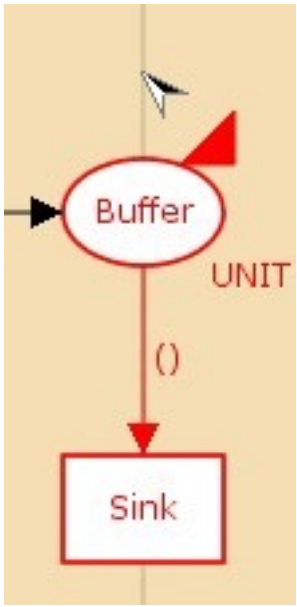
You can use the tools from the Style tools to change the graphical attributes (color, thickness and line-pattern) of objects.
The Style tools contains these tools:

- Color tools;
- Set Line Thickness;
- Cycle Arc Head Size;
- Toggle Fill;
- Clone Style.

To change the color, line thickness, or line style of an object, pick up the tool by clicking on its cell and then click on the object you want to apply the tool to.
If the tool is a cycle tool, such as Cycle Arc Head Size, click several times until the object has the property you want.







If several objects are placed on magnetic guidelines, you can apply the style tool to the guideline to apply it to all objects on the guideline.

If you want more information about using these tools, then please visit the Style tools page.