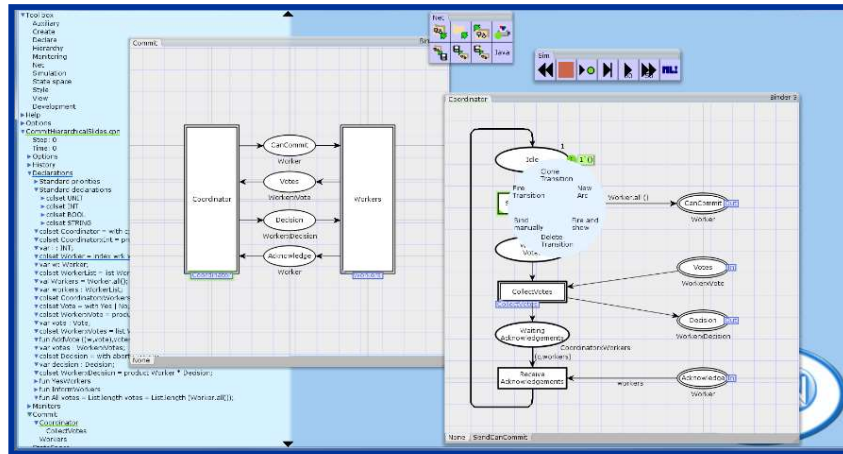


# Coloured Petri Nets



# Lars M. Kristensen

# Department of Computing, Mathematics, and Physics

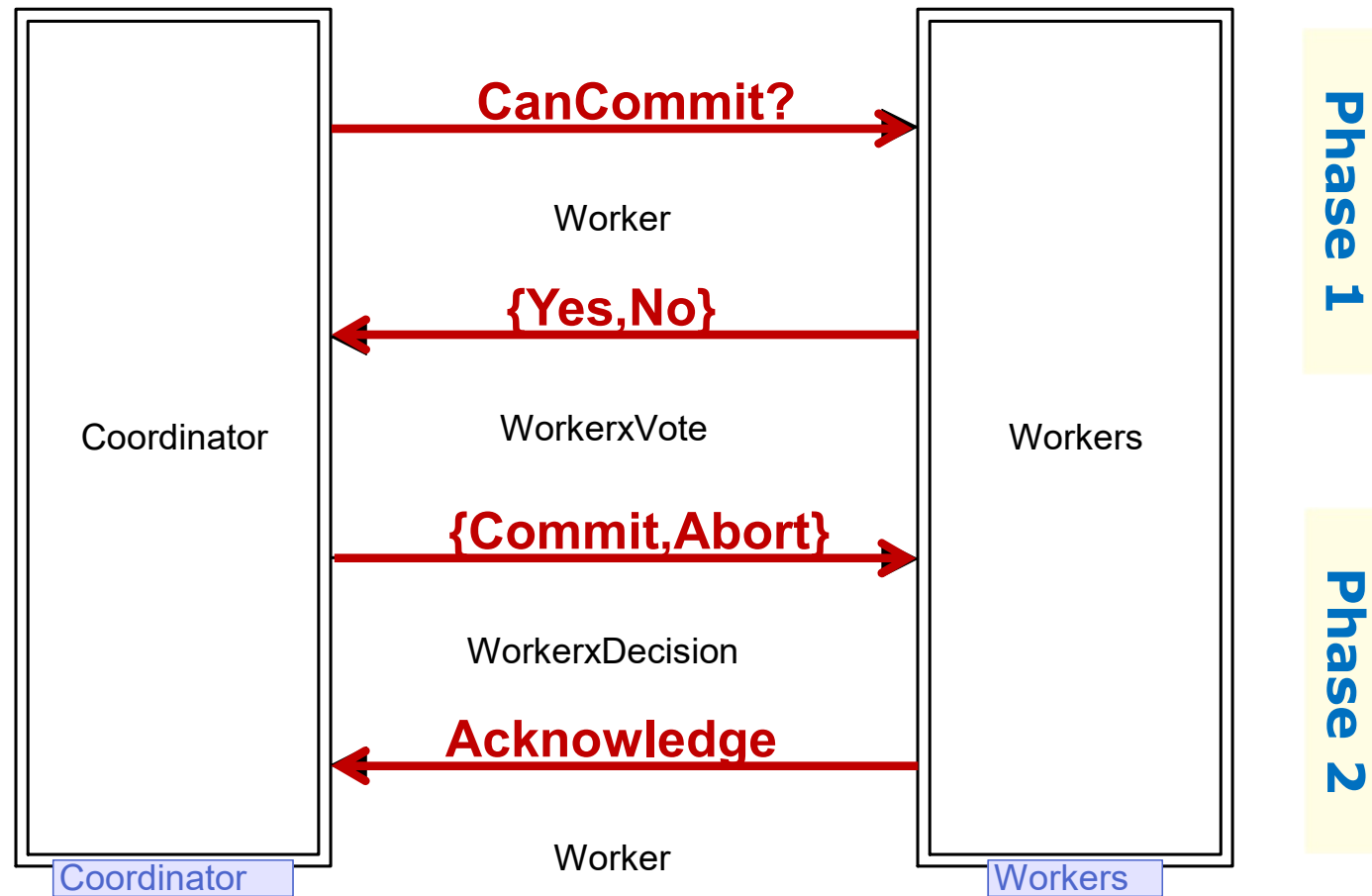
# Western Norway University of Applied Sciences

**Email:** [lmkr@hvl.no](mailto:lmkr@hvl.no) / **WWW:** [home.hib.no/ansatte/lmkr](http://home.hib.no/ansatte/lmkr)

# Introduction

- **Address the practical shortcomings of PT-nets.**
- **Coloured Petri Nets (CPNs) = PT-nets + Standard ML programming language**
  - Places may have a **type** and tokens can carry **data values**
  - Transitions may have **variables** that can be bound to values
  - **Arc expressions** determine the tokens added/removed
  - **Guard expressions** may be used as an extra enabling condition
- **Standard ML = functional programming**
  - Computation proceeds by **evaluation of expressions**
  - **Static typing** with the type of expressions being **inferred**
  - **Functions** are first-order values and can be polymorphic
  - **Recursion** and lists are used to express iteration

# Two-phase Commit Transaction Protocol



# Colour Set Definitions

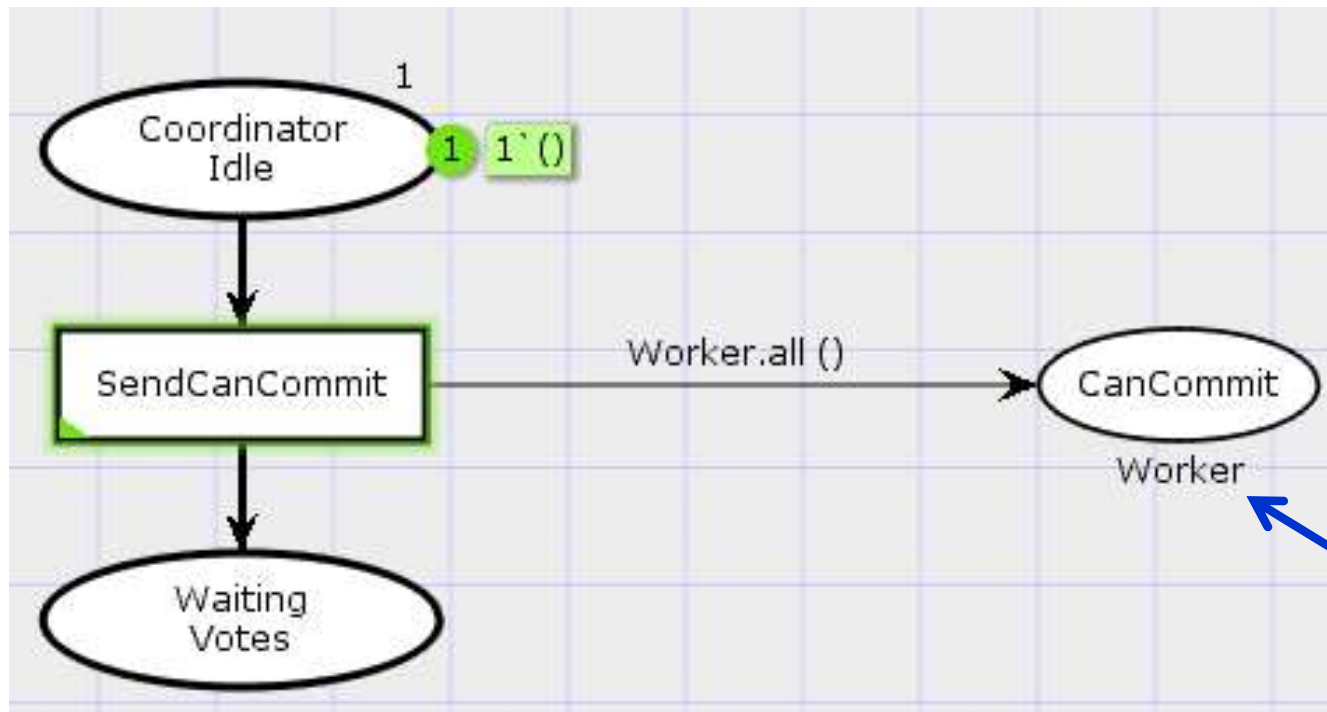
- Determines the **data types** that can be used in the model

Colour set definitions	Example values
<code>val W = 2;</code>	<code>wrk(1), wrk(2)</code>
<code>colset Worker = index wrk with 1..W;</code>	<code>Yes, No</code>
<code>colset Vote = with Yes   No;</code>	<code>(wrk(1), Yes)</code>
<code>colset WorkerxVote = product Worker * Vote;</code>	<code>Abort, Commit</code>
<code>colset Decision = with Abort   Commit;</code>	<code>(wrk(1), Commit)</code>
<code>colset WorkerxDecision = product Worker * Decision;</code>	

- Also colour set constructors for lists (**list**), records (**record**), and unions (**union**)
- **Base data types:** **UNIT**, **INT**, **STRING**, **BOOL**, **REAL**

# Colour Set of a Place

- Determines the **kinds of tokens** that may reside on the place



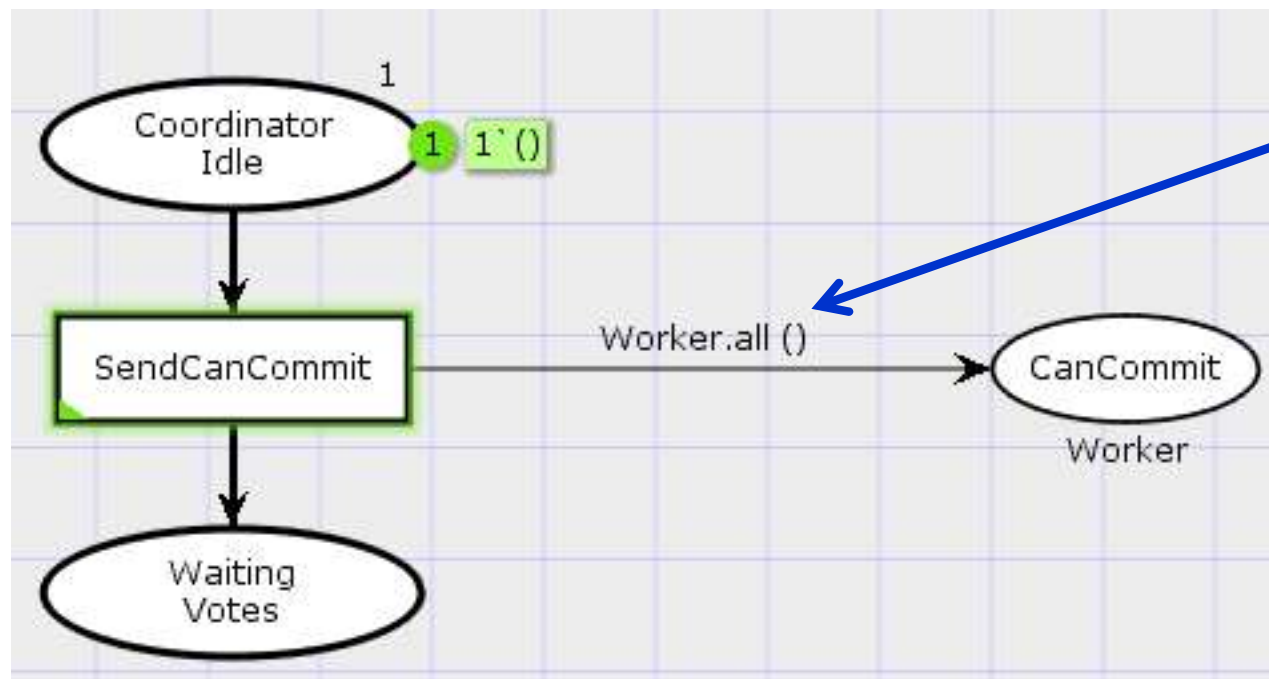
CoordinatorIdle and WaitingVotes are ordinary places

Tokens on CanCommit can have the values `wrk(1)` and `wrk(2)`

The colour set is by convention written below the place

# Arc Expressions

- Determine the tokens that are removed/added from/to places when transitions occur



Expression evaluating to all values in the Worker colour set

`Worker.all ()`



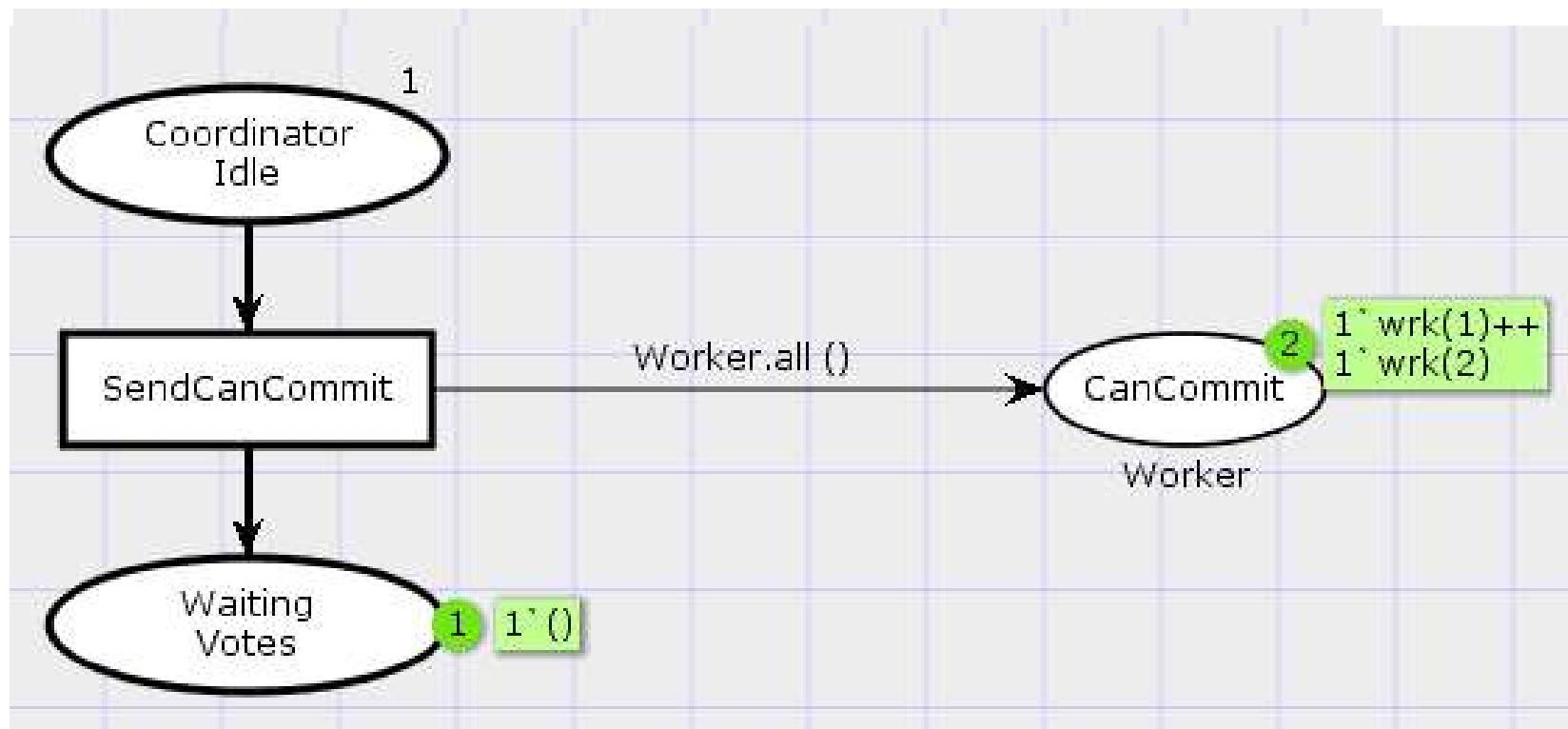
evaluate

`[ wrk(1), wrk(2) ]`

- The type of an arc expression **must match** the colour set of the place connected to the arc.

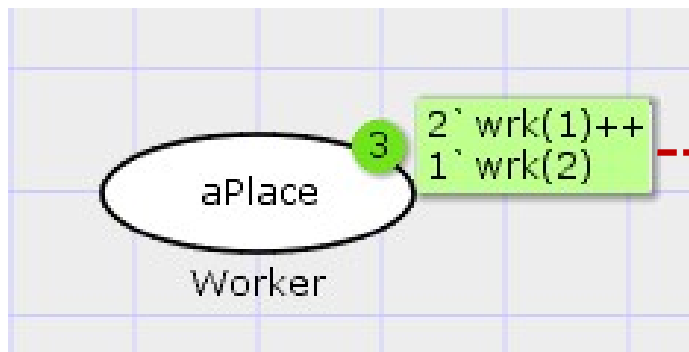
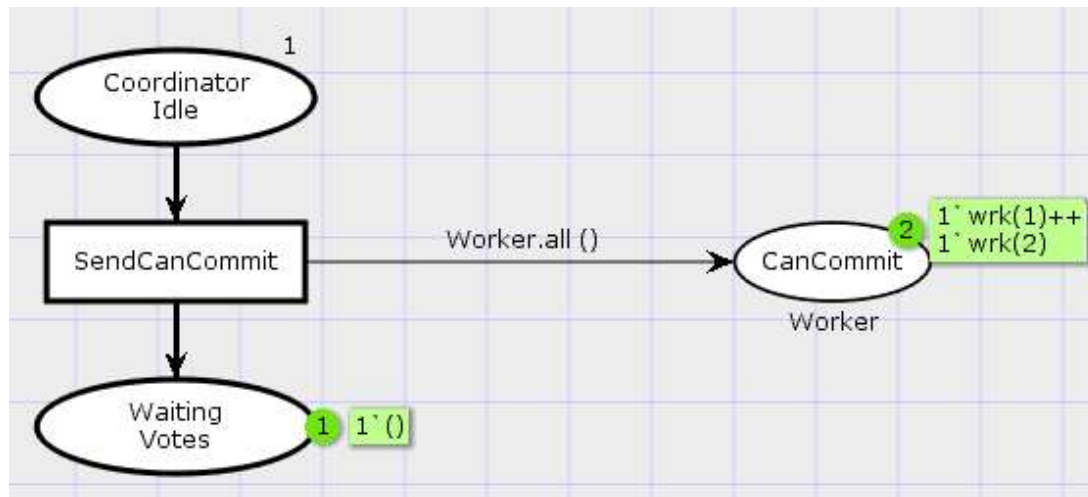
# Evaluation of Expressions

- The tokens added and removed are determined by **evaluating arc expressions**



# Markings and Multi-sets

- Each place may hold a **multi-set of tokens** over the colour set of the place



## Multi-set notation

coefficient («of»)

token colour  
(value)

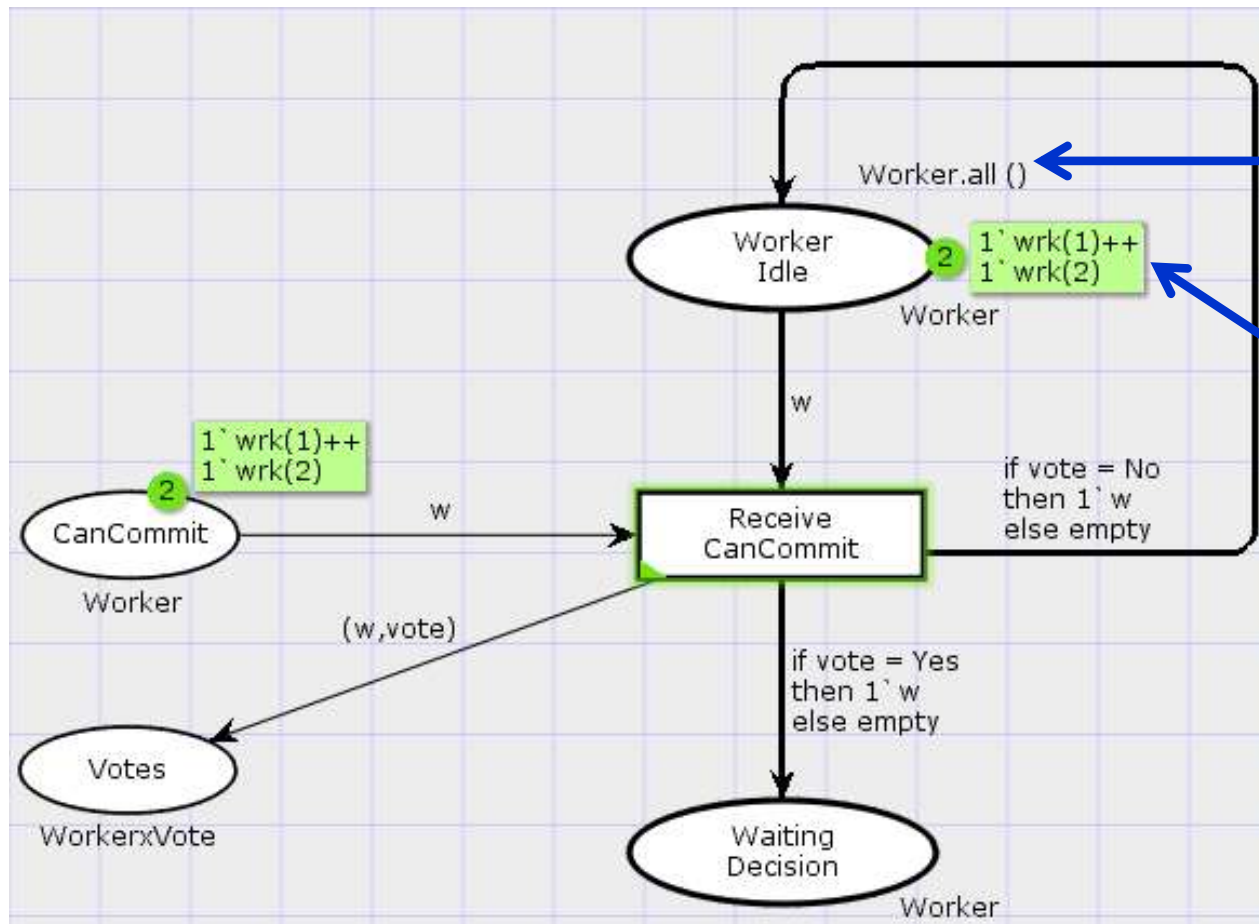
2 `wrk(1) ++  
1 `wrk(2)

union («and»)



# Initial Marking

- The **initial marking** (state) is obtained by evaluating the **initial marking expressions**

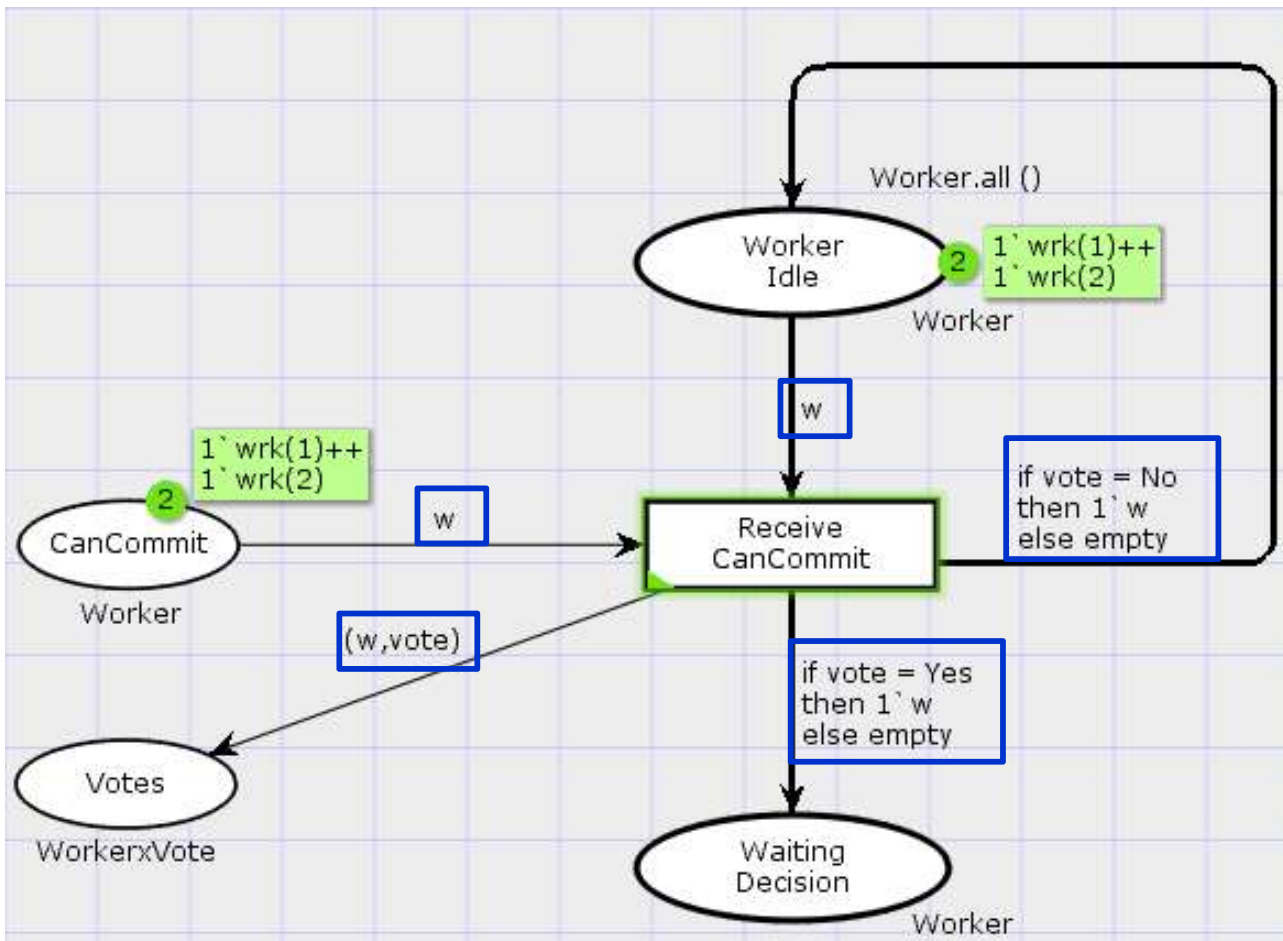


The initial marking is by convention written above the place

The two workers are initially **Idle**

# Transition Variables

- The arc expressions on the arcs of a transition may contain **free variables**



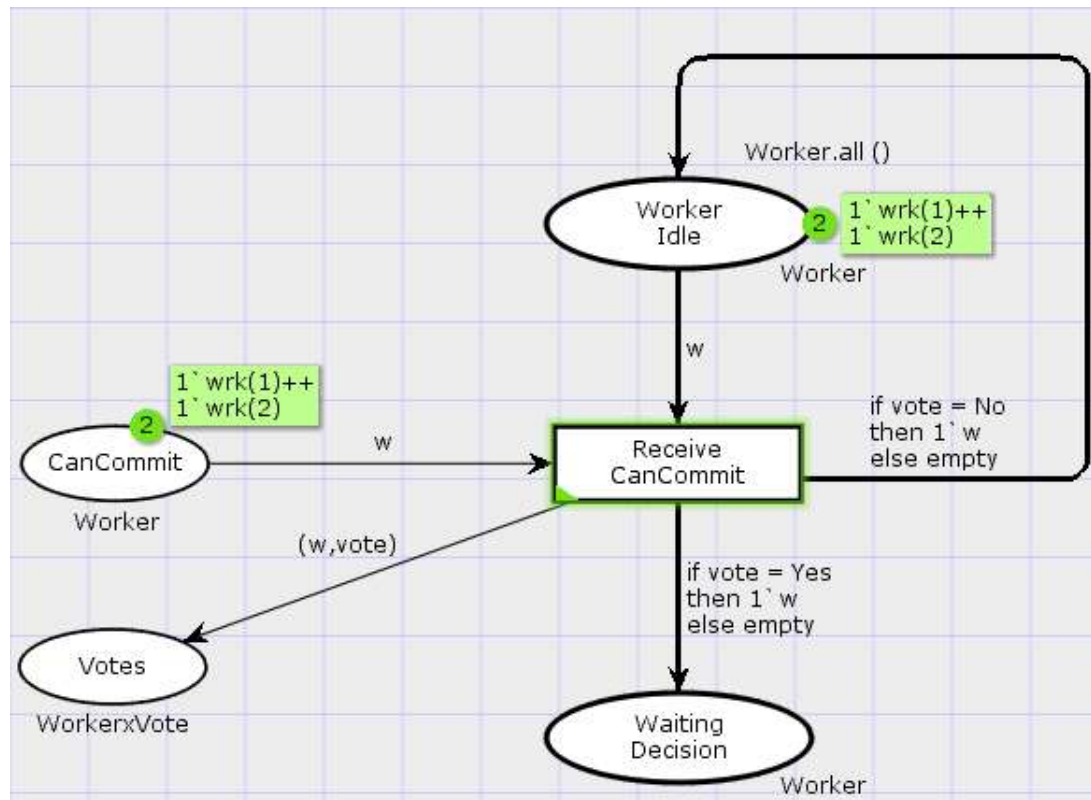
## Variable declarations

```
val W = 2;  
colset Worker =  
    index wrk with 1..W;  
var w : Worker;  
  
colset Vote = with Yes | No;  
var vote : Vote;
```

Arc expressions with free variables **vote** and **w**.

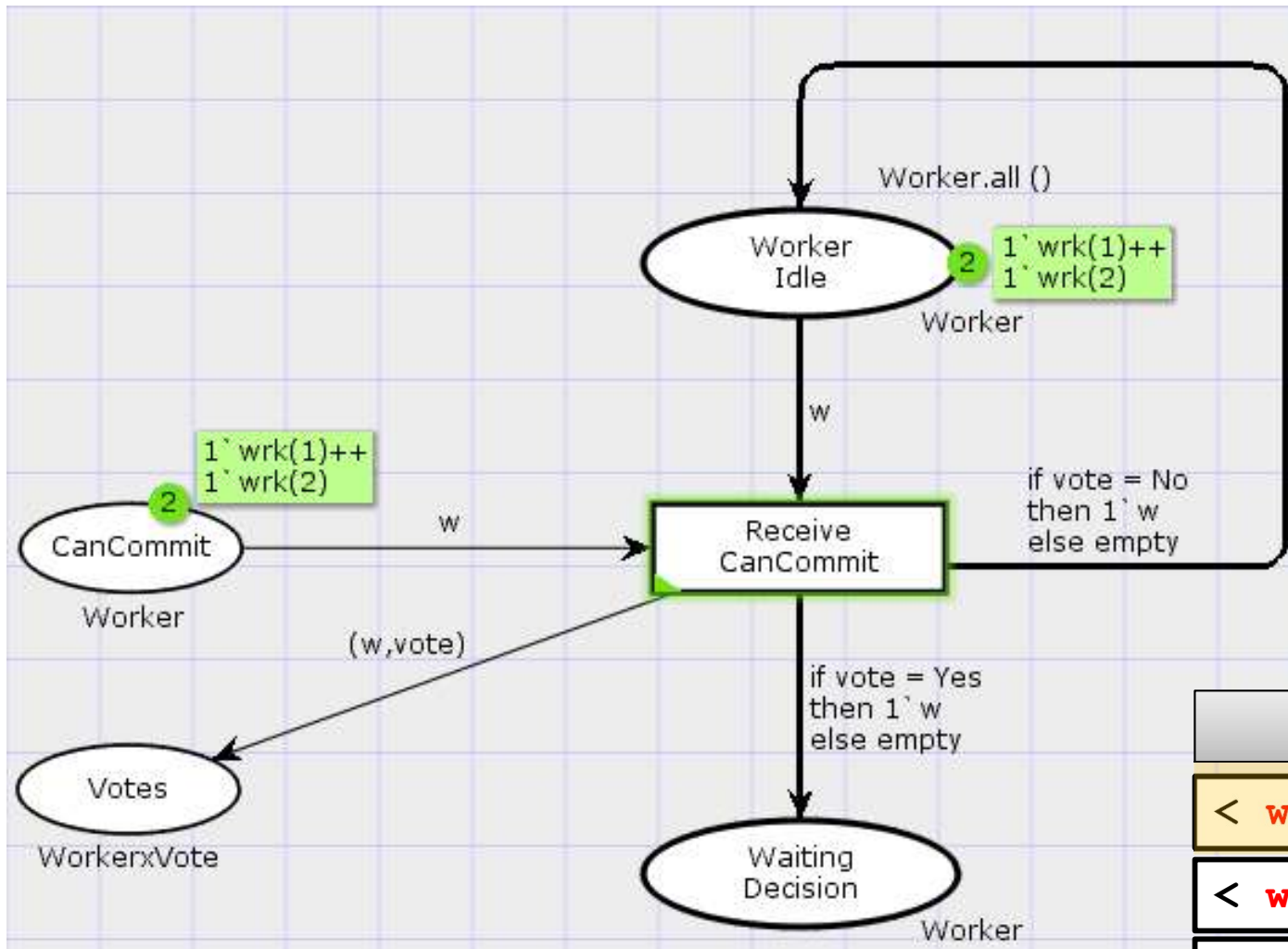
# Transition Variables

- Transition **ReceiveCanCommit** has two free variables: **vote** and **w**



- Variables must be **bound** to values for a transition to be enabled and occur
- Similar to formal and actual parameters known from programming
- The association of values to variables is called a **binding**.
- The bindings correspond to possible **enabling** and **occurrence modes** of the transition.

# Transition Bindings



## Possible bindings ?

**< w = wrk(1), vote = Yes >**

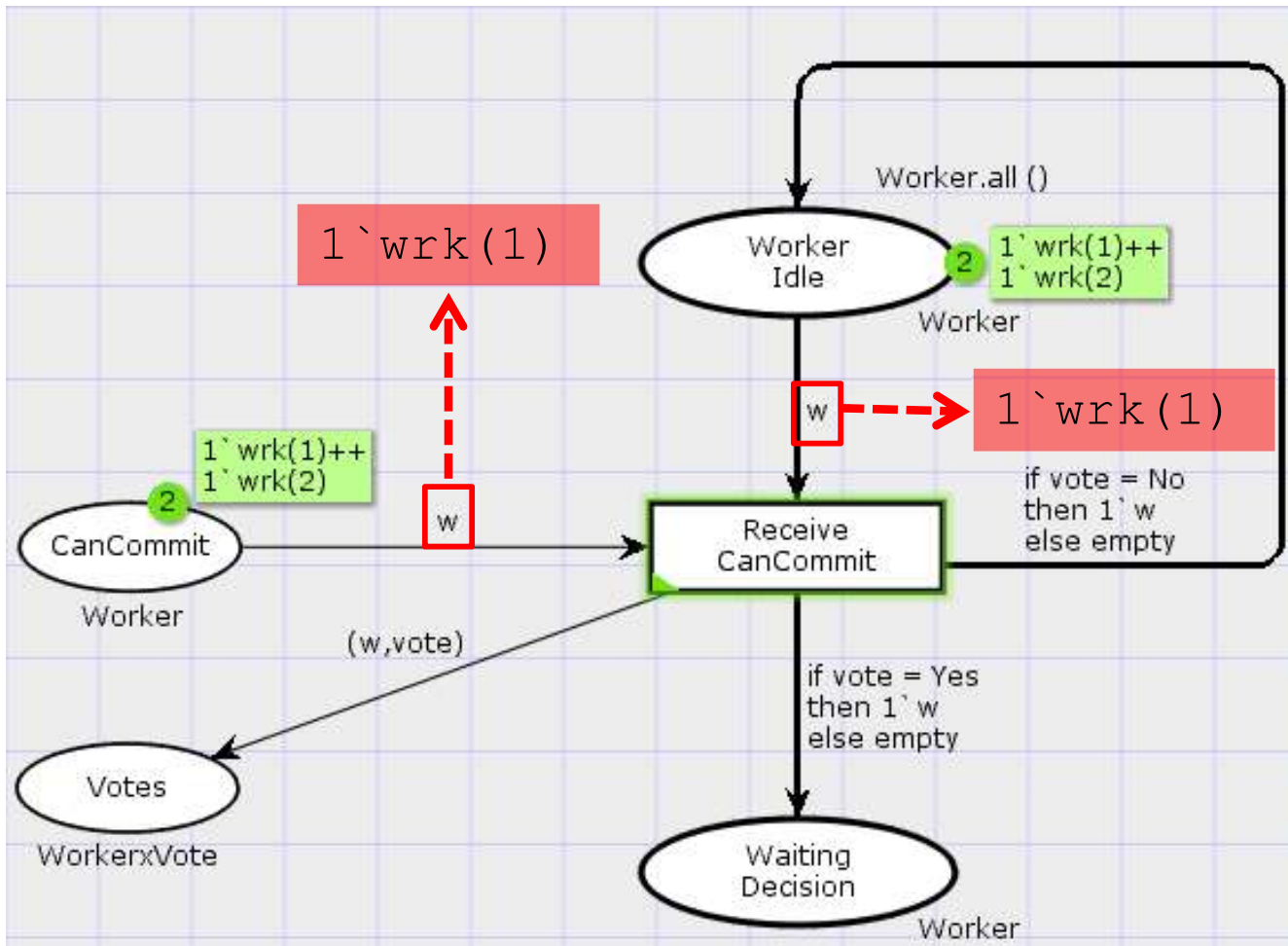
**< w = wrk(1), vote = No >**

**< w = wrk(2), vote = Yes >**

**< w = wrk(2), vote = No >**

# Binding Enabling

$\langle w = wrk(1), vote = Yes \rangle$

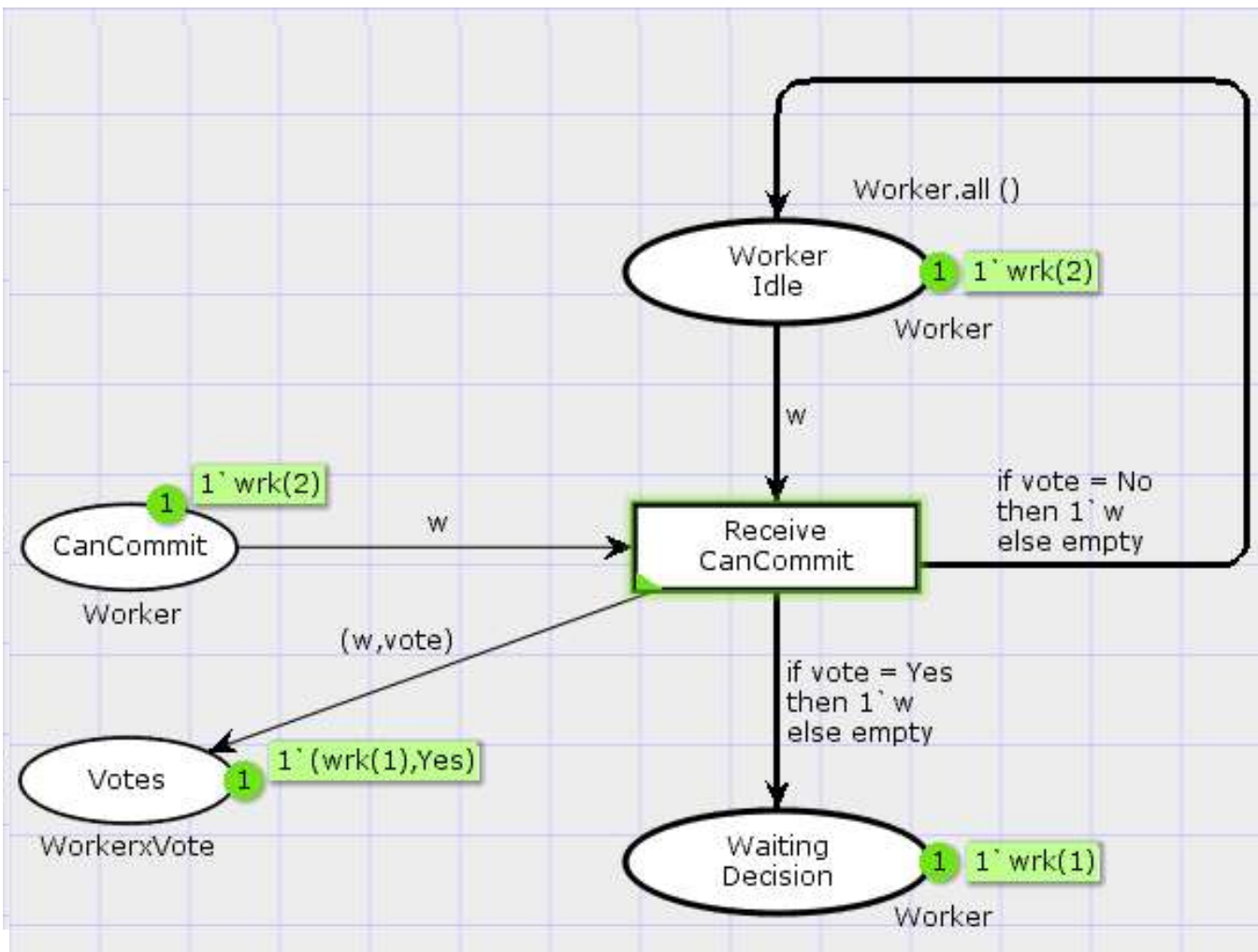


- A transition binding is **enabled** if there are sufficient tokens on each input place.
- **Tokens required on input places** are determined by **evaluating the input arc expressions** in the **binding** under consideration.
- **Enabling condition:** the multi-set of tokens obtained must be **contained in** the multi-set of tokens present on the corresponding input place.



# Binding Occurrence

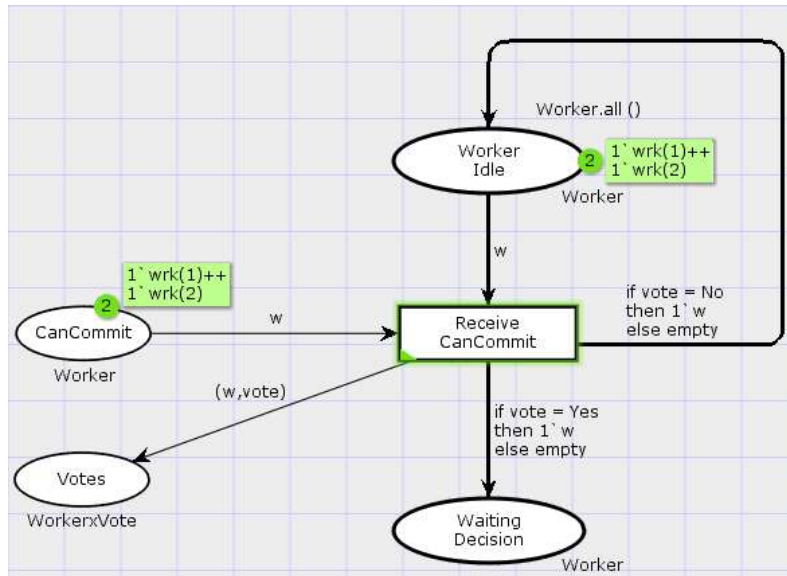
$\langle w = \text{wrk}(1), \text{vote} = \text{Yes} \rangle$



- An enabled transition binding may **occur** changing the current marking (state)
- **Tokens removed from input places:** determined by evaluating the input arc expression in the binding.
- **Tokens added to output places:** determined by evaluating the output arc expressions in the binding.

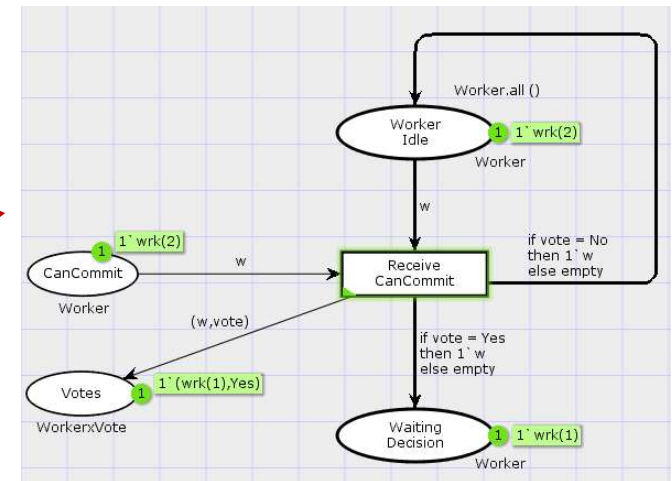
# Binding Occurrence

- A transition may have several enabled bindings



$b_{1Y}$

$b_{1N}$



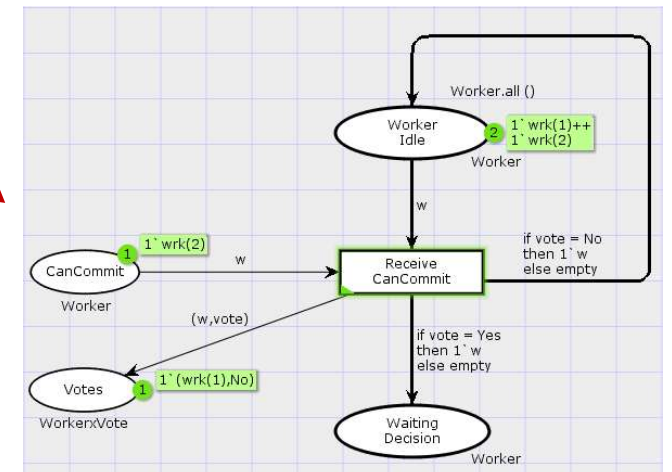
## Bindings

$b_{1Y} = \langle w = wrk(1), vote = Yes \rangle$

$b_{1N} = \langle w = wrk(1), vote = No \rangle$

$b_{2Y} = \langle w = wrk(2), vote = Yes \rangle$

$b_{2Y} = \langle w = wrk(2), vote = No \rangle$



# CPN Tools Demo

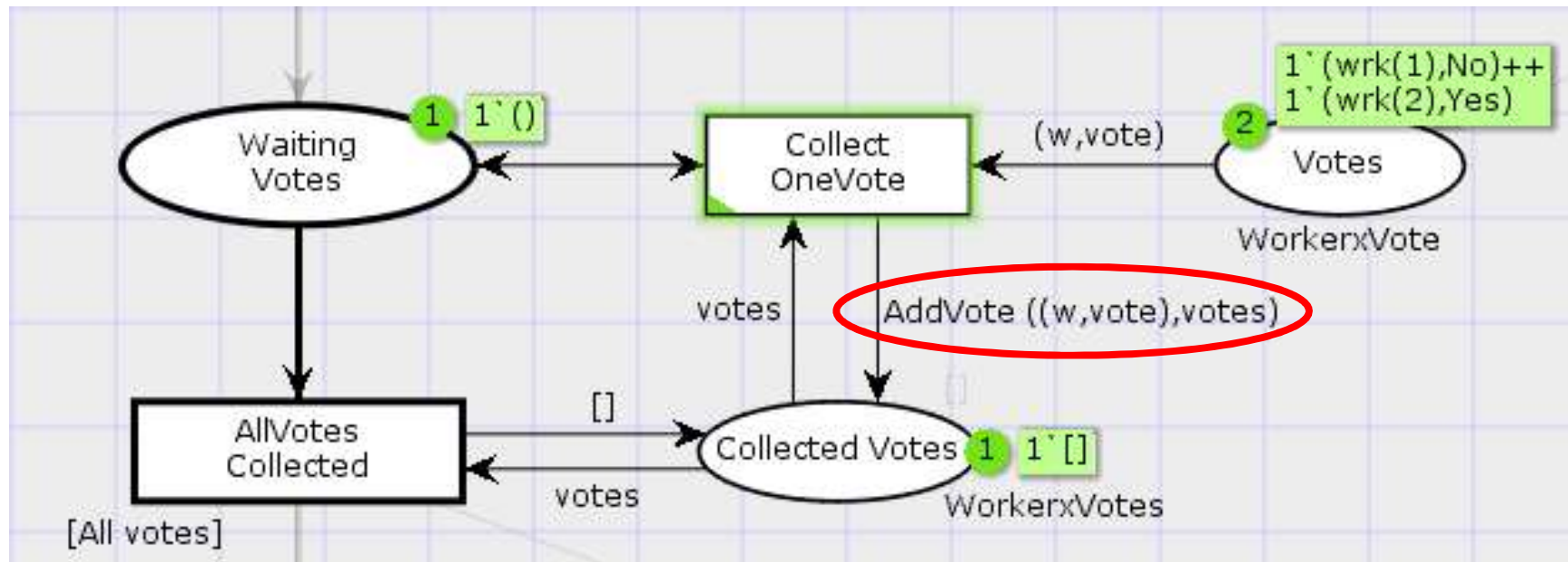
- **Simulation of CPN models**
  - Interactive simulation with binding selection
  - Returning to the initial marking
  - Automatic simulation with visual feedback
  - Stop options and automatic simulation





# Collecting Votes

- Votes are collected one at a time and accumulated in a list-token on place **Votes**



```

var w      : Worker;      var vote : Vote;
colset WorkerxVote = product Worker * Vote;
colset WorkerxVotes = list WorkerxVote;

var votes : WorkerxVotes;
    
```

# Functions

- The function **AddVote** is used to add a vote from a worker to the list of collected votes

```
fun AddVote ((w,vote),votes) = (w,vote)::votes
```

- **Example**

```
AddVote ((wrk(1),Yes),[])
```



```
(wrk(1),Yes)::[]
```



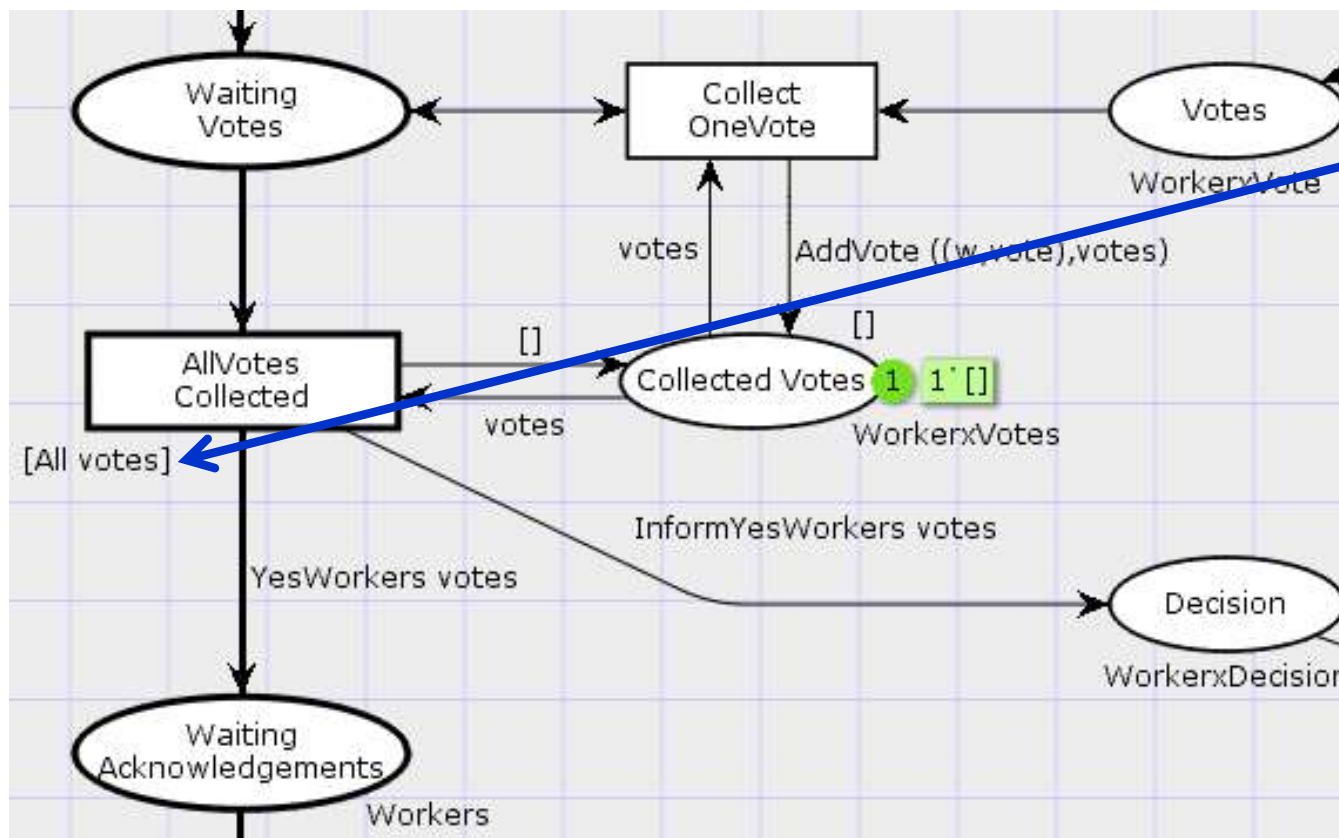
```
[(wrk(1),Yes)]
```



**:: is the list-constructor  
in Standard ML**

# Guard Expressions

- A transition may have a boolean guard expression which is extra enabling condition



The guard is by convention written in square brackets next to the transition

```
fun All votes =
    (List.length votes = W)

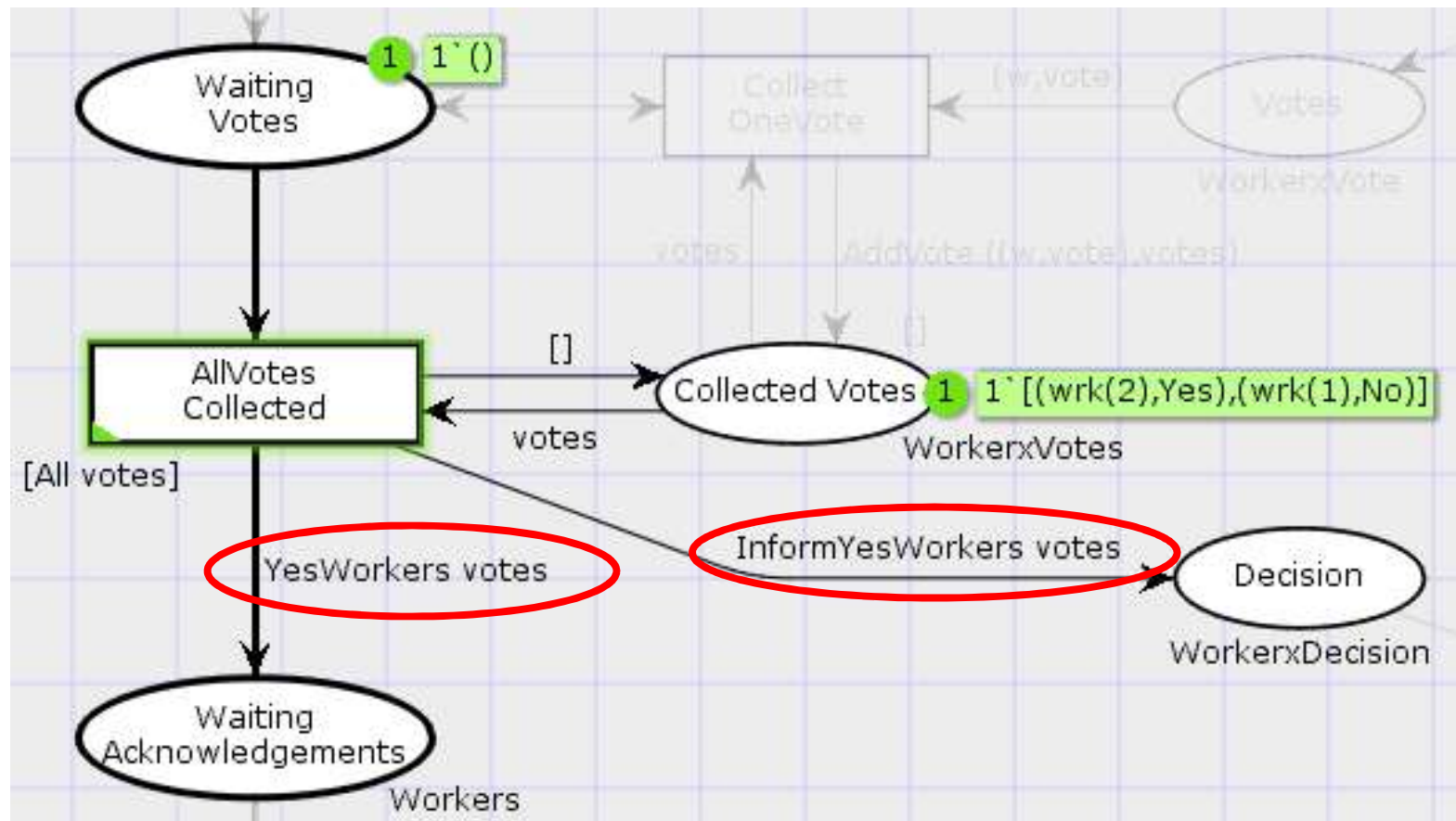
var votes : WorkerxVotes

colset WorkerxVote =
    product Worker * Vote;

colset WorkerxVotes =
    list WorkerxVote;
```

# Informating Workers

- Functions are also used to find the workers that needs to be informed about the decision



# Higher-order Functions

- **Obtaining the list of Yes-votes**

```
fun yesVotes votes = List.filter  
                        (fn (w,vote) => vote = Yes)  
                        votes
```

```
yesVotes [ (wrk(1), No), (wrk(2), Yes) ]
```



```
yesVotes [ (wrk(2), Yes) ]
```

- **Getting workers that votes Yes (projection)**

```
fun YesWorkers votes = List.map  
                        (fn (w,_) => w)  
                        (yesVotes votes)
```

# Higher-order Functions

```
fun InformYesWorkers votes =  
  let  
    val yesworkers = YesWorkers votes  
    val decision =  
      (if (List.length yesworkers = W)  
        then commit  
        else abort)  
  in  
    List.map (fn w => (w,decision)) yesworkers  
end
```

InformYesWorkers [ (wrk(2), Yes) ]



[ (wrk(2), abort) ]

# CPN Tools Demo

- **Editing of CPN models**

- Incremental syntax check of the model (dependencies)
- Adding and deleting declarations
- Editing inscriptions  
(arc expressions, colour sets, initial markings, guards)
- Guidelines, graphical attributes, and groups



# Summary

- **Coloured Petri Nets extends Place/Transition Nets with a functional programming language.**
- **Syntactical concepts**
  - **Colour sets** defines the data types that can be used
  - Declaration of **variables**
  - Places have a colour set determining the **kind of tokens** a place may contain
  - Arc, initial marking, and guard expressions
- **Semantical concepts**
  - The marking of a place is a **multi-set of token values**
  - A **binding** gives values to the variables of a transition
  - **Evaluation of arc expressions** in bindings determines the tokens remove/added by transitions.