TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Rendezvous-Multiple Broadcast Networks

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Masterstudium Technische Mathematik

Schwerpunkt Analysis und Geometrie

eingereicht von

Johannes Hafner BSc
Matrikelnummer 00927401

ausgeführt am Institut für Logic and Computation
der Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Associate Prof. Dipl.-Math. Dr.techn. Florian Zuleger
Mitwirkung: Univ.-Ass. Benjamin Aminof PhD

Wien, 25.10.2018

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Abstract

Automatic software verification is concerned with automatically deciding whether a given model of a computer system satisfies a given specification. The Parameterized model checking problem (PMCP) of networks is the special case of deciding whether a given network (usually of identical processes) satisfies its specification regardless of the number of processes in it. We explore the case where the processes are indeed all identical, and communicate via rendezvous and symmetric broadcast. We map the boundary of decidability of the PMCP in such networks. In particular, we show that the PMCP is decidable for finite runs of such networks, whereas for infinite runs the situation is more involved: the PMCP is undecidable already for networks with only two types of broadcast messages (the case of a single symmetric broadcast is already known to be decidable), but becomes decidable if one introduces certain grouping structures limiting the scope of broadcast messages.

Contents

CHAPTER 1

Introduction

Automatic software verification is the art of writing computer programs capable of deciding whether a given model of a software or hardware system behaves according to a given specification. These specifications consist of questions like if a property is always true for a run of the system. Alternatively, it may be of relevance how often or how frequent a situation can occur.

One class of systems of interest are networks of identical processes, each running the same code (represented by a *process template*). There are many types of such networks that are achieved by varying one of two aspects: the topology of the network and the communication protocols. The topology describes the connectivity of the network in the sense of which processes can directly communicate with which other processes. The communication protocols govern how processes communicate with each other, and one of the most important aspects of this are the form of message between the processes (e.g., broadcast, or direct message passing between two processes).

We will focus on networks with a clique topology, i.e., ones in which each process can communicate directly with every other process. Note that this is meant not at the physical level, but at the more abstract high level. For example even though not every computer on the Internet is directly physically connected to every other computer, it is still considered a clique topology for our purposes since every computer can direct a message to any other computer of its choice.

In terms of communication, we will focus on networks where processes communicate using two types of communication primitives: *symmetric broadcast* and *rendezvous*. In asymmetric broadcast, where one process sends a message to all other processes, and the sender is potentially distinguished from the receivers. In symmetric broadcast however, sending a broadcast message has the same effect on the sender as on any receiver. It therefore makes sense to think of symmetric broadcasts as all processes receiving the message and leaving open where the message comes from.

Rendezvous messages are sent from one process to a fixed number of other processes. Where there are only two processes involved one usually speaks of *pairwise rendezvous*, and when k processes are involved one speaks of *k-wise rendezvous*. The source of the name 'rendezvous' comes from the fact that one distinguishes between not only sender and receivers, but between each one of the k processes involved in a k-wise rendezvous. I.e., two receivers that are in the same state may behave differently since they are assigned different roles in the rendezvous. Thus, it is common not to think of senders and receivers also in this case, and simply consider that k processes 'rendezvous' and change states according to the k different roles they assumed during this meeting. It is worth noting also that the state a process is in determines which (if any) rendezvous it can participate in, and in which capacity. On the other hand every process can receive a broadcast message regardless of the state it is in.

The networks we investigate are thus called *Rendezvous-Multiple Broadcast Networks* (RMBNs for short), and we study them in general as well as focus on some special cases where additional restrictions (or structure) are imposed on the communication primitives.

1.1 Problem Statement

We are interested in the behaviour of individual processes in RMBNs.
Specifically, we examine the *Parametrized Model Checking Problem (PMCP)*: Given a template describing each processes in the network, and a specification for the behavior of a single process, decide whether all processes conform to the specification regardless of the the size of the network (i.e., for networks with any number processes).

1.2 Aim of the Work

Are work is concerned with mapping the boundary of decidability of the PMCP problem. Naturally, the usual tradeoff exists: the more powerful the system model is the more likely is PMCP to be undecidable; and so it is in our case. below we describe the state of the art before this thesis and our contributions.

1.3 State of the art

Early positive results for networks were given by [CGB86] and [GS92]. [CGB86] showed ways to conclude properties of networks with any number of processes from networks with two processes. [GS92] is working with rendezvous networks. In contrast to to the networks in this thesis, they do not use broadcast transitions. They look at networks with and without a controller process. A controller process is a unique type of process that only appears once in the network.

It was shown in [EFM99] that for networks with rendezvous and asymmetric broadcasts, the PMCP is decidable for finite executions but undecidable for infinite executions if there are at least two different messages used for asymmetric broadcasts.

In [ARZS14] it was shown that the PMCP is decidable for finite and infinite runs in networks with rendezvous and only one type of symmetric broadcast. The languages created by infinite executions of individual processes in these networks are ωB-regular. In [ARZS14], the broadcast is an abstraction of ticks of a discrete clock.

While this proves the decidability of networks with discrete clocks, the problem is harder for real-valued clocks. The question whether a given state can be visited infinitely many times is undecidable with one real-valued clock in each process [AJ03]. For finite runs however, the PMCP is decidable for one real-valued clock. With two real-valued clocks per process, the PMCP is not even decidable for finite runs [ADM04].

The networks in [EFM99], [AJ03] and [ADM04] have a controller process, while [ARZS14] and the networks here do not have a controller process, i.e. all processes run the same code.

1.4 Methodological approach

Generalizing the model in [ARZS14], we look at different rendezvous-broadcast networks with multiple broadcast messages. Undecidability results are created via reduction to known undecidable problems, building on the results in [EFM99]. Decidability results are created by outlining an algorithm constructing an automaton that recognizes the language of all executions of a given network. [ARZS14] has outlined how this can be used to solve the PMCP.

In our search for decidable models, we introduce a new kind of broadcast messages: Group broadcasts. They can appear in networks where the processes are assigned to groups. A group broadcast message is received by all processes in one group.

1.5 Outline and Results

- Chapter 2 gives an overview of notation and tools used.
- Chapter 3 gives results for general rendezvous-multiple broadcast networks (RMBN) with symmetric broadcasts. While the PMCP is decidable for finite runs, infinite runs are already undecidable with two types of broadcast messages.
- Chapter 4 introduces the rendezvous-fixed group broadcast network (RFGBN) model. This model has a fixed number of groups built in, which can communicate via group broadcast. Both the decidability result for finite runs and the undecidability result for infinite runs can be extended to RFGBN from RMBN.

- Chapter 5 introduces the rendezvous-group broadcast network (RGBN) model. In this model, processes get assigned to any number of groups at the beginning of the run. In this case, the PMCP is decidable for both finite and infinite runs.
- Chapter 6 generalizes the RGBN by allowing processes to change group as a consequence of rendezvous messages. The PMCP is still decidable for both finite and infinite runs.

CHAPTER 2

Definitions and Preliminaries

In this section we will go over some definitions and notations used throughout all sections.

2.1 General notations

$\mathbb{N}$ denotes the set of non negative integers. For $n \in \mathbb{N}$ let $[n]$ refer to $\{1, \ldots, n\}$, the set of the first n positive integers. For two sets $\mathcal{X}$ and $\mathcal{Y}$, let $\mathcal{X}^{\mathcal{Y}}$ be the set of all function $f : \mathcal{Y} \to \mathcal{X}$. If π is a sequence and $i \in \mathbb{N}$, let π_i refer to the i-th element of the sequence and $|\pi|$ to the length of the sequence, which can be ∞. If $\vec{s}$ is an N-dimensional vector and $n \in [N]$, let $\vec{s}(n)$ refer to the n-th element of the vector. Let $dim(\vec{s}) = N$ be the dimension of the vector. For a positive integer N and a set $\mathcal{X}$ let $\mathcal{X}^N$ be the set of N-dimensional vectors over the set $\mathcal{X}$.
For a set $\mathcal{X}$ and two integers N, M let $\mathcal{X}^{N \times M}$ be the set of $N \times M$ matrices with values in $\mathcal{X}$. For three sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, let $\mathcal{X}^{\mathcal{Y} \times \mathcal{Z}}$ be the set of $|\mathcal{Y}| \times |\mathcal{Z}|$ matrices with values in $\mathcal{X}$, row names in $\mathcal{Y}$ and column names in $\mathcal{Z}$. For $X \in \mathcal{X}^{N \times M}$ and $n \leq N$, $m \leq M$, $X(n, m)$ denotes the element in row n and column m. For $X \in \mathcal{X}^{\mathcal{Y} \times \mathcal{Z}}$, $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$, $X(y, z)$ denotes the element in row with name y and column with name z.
For three sets $\mathcal{X}, \mathcal{Y}$, $\mathcal{Z}$ and two functions or vectors $f : \mathcal{Y} \to \mathcal{Z}$ and $g : \mathcal{X} \to \mathcal{Y}$, let $f \circ g : \mathcal{X} \to \mathcal{Z}$ be the concatenation of f and g, that is the function or vector $f \circ g(x) = f(g(x))$. $\circ$ will also be applied to vectors.

2.2 Labeled Transition Systems

In this thesis, all networks and the templates they are defined with are a special case of a Labeled Transition System (LTS).

Definition 1. Labeled Transition System

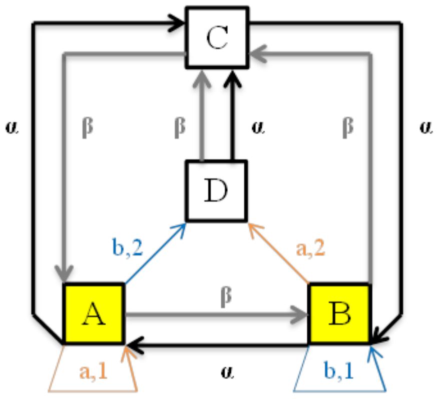

Figure 2.1: Example of a rendezvous-multiple broadcast network with 2 broadcast types.

An LTS is an quadruple $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{S}_0$ is a set of initial states, $\mathcal{A}$ is a set of labels and $\mathcal{T}$ is a set of transition. The transitions are triples and elements of $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$. For $s, s \in \mathcal{S}$ and $a \in A$, the transitions are usually written in the form $s \xrightarrow{a} s'$. For such a transition t, we denote the following access functions:

- $src(t) = s$, the *source* of the transition
- $dst(t) = s'$, the *destination* of the transition
- $label(t) = a$, the *label* of the transition

A sequence of transitions π is called a *path* of the LTS, if for all $i < |\pi|$ $dst(\pi_i) = src(\pi_{i+1})$, i.e. the destination of each transitions must be the source of the next transition.
We write $src(\pi) = src(\pi_1)$ for the source of the path and for a finite path $dst(\pi) = dst(\pi_{|\pi|})$ denotes the destination of the path.
A path is called a *run* of the LTS, if $src(\pi) \in \mathcal{S}_0$. In some special cases of LTS, there are additional acceptance conditions for a path to be called a run.

2.3 Example Network

Figure 2.1 gives an example of a template for a network. It describes a rendezvous-multiple broadcast network, which will be formally defined in Chapter 3.

States are displayed as squares with a capital letter in them. Initial states have yellow background. Broadcast transitions are bold and labeled with a Greek letter. There are two broadcast letters α and β and broadcast transitions with the same letter have the same color. Rendezvous transitions are represented by thin arrows. For this template, two process take each rendezvous transition together. There are two rendezvous letters a and b. There are two roles for both letters, 1 and 2. There must be a process in both

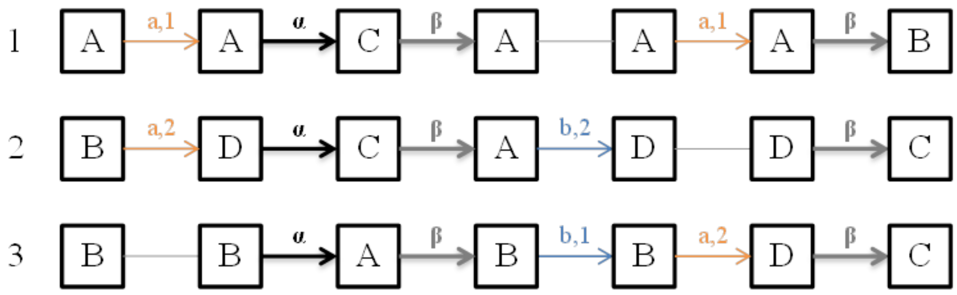

Figure 2.2: Example run of the network described by the template in Figure 2.1

source states A and B in order for a global rendezvous transition labeled with either a or b to happen.

Figure 2.2 gives an example of how a run in the network described in Figure 2.1. It has three processes. Each line of transitions gives the transitions each of the Processes 1, 2 and 3 takes. Initially, Process 1 is in State A, while Processes 2 and 3 are in State B. From left to right, the following global transition happen: First, there is a rendezvous transition with letter a, where Process 1 takes Role 1 and Process 2 takes Role 2. That transfers Process 2 into state D, while keeping Process 1 in State A. Process 3 is inactive in this transition. This is displayed by a thin gray line and keeps Process 3 in State B. Next comes a broadcast transition labeled with α. Here, every Process follows the unique transition labeled with α starting in the State it is currently in.
After that there is:

- A broadcast transition with label β
- A rendezvous transition with label b, where Process 3 takes Role 1 and Process 2 takes role 2
- A rendezvous transition with label a, where Process 1 takes Role 1 and Process 3 takes role 2
  and finally another broadcast transition with label β

In the end, Process 1 is in State C and Processes 2 and 3 are in State C.

2.4 Grammars and Automata

The specifications for the PMCP describe a language, either with grammars or automata. For finite words, the specifications must be regular languages. Regular languages are recognized by finite word automata [MS97] and regular grammars. A regular grammar is recursively defined and is of the forms:

expression	meaning
ϵ	The empty word
ab	Word recognized by a followed by a word recognized by b
$a \cup b$	Word recognized by a or by b
a^*	Word consisting of any finite number of words all recognized by a

where a and b are both regular expressions themselves.

ω-regular languages are the equivalence of regular languages for infinite words. They are sets of infinite words recognized by a grammar following the construction principles:

expression	meaning
ϵ	The empty word
ac	Word recognized by a followed by a word recognized by c
a^ω	Word consisting of an infinite sequence of words all recognized by a

where a is a regular expression and c is an ω-regular expression.
As parts of ω-regular expressions are regular, they also contain Kleene stars * and $\cup$.

The automata and networks in this thesis show behavior that can not be described by ω-regular languages as they are extensions of the networks presented in [ARZS14]. Therefore, ωB-regular languages are used. They possess ^{B}, in addition to the Kleene star *. The number of repeats of the word recognized by a^B must be bounded in the infinite word. The expression $(\alpha^B\beta)^\omega$ for example recognizes every infinite word over the alphabet $\{\alpha, \beta\}$ with an infinite number of β's and where the number of α's between two consecutive β's is bounded by any natural number.

B-automata recognizing ωB-regular languages are equipped with counters, which can be increased and reset. A run is accepted if the counters stay bounded.
Every ω-regular language is also ωB-regular but not the other way round.

B-automata were introduced in [BC06]. A equivalent definition for B was simultaneously developed in [AKY08]. The formulation here is based on an equivalent formulation in [Boj10].

Definition 2. B-automata

Given a set of counters $\mathcal{C}$ and a set of letters $\mathcal{A}$, a B-automaton is an LTS extended by an acceptance condition $B = \left\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times \left(((\mathcal{C} \cup \{0\}) \times \mathbb{N})^{\mathcal{C}}\right), \mathcal{T}, \Phi \right\rangle$. The transitions are written in the form $s \xrightarrow[\overline{c}]{a} s'$ with $a \in \mathcal{A}$ and $\overline{c} \in ((\mathcal{C} \cup \{0\}) \times \mathbb{N}$
If $\overline{c}(\gamma) = (0, k)$, $\overline{c}(\delta) = (\beta, k')$ with $k, k' \in \mathbb{N}$ and $\gamma, \delta, \beta \in \mathcal{C}$ and $\overline{c}(\alpha) = (\alpha, 0)$ for all other counters $\alpha \in \mathcal{C}$, then the transition is written as $s \xrightarrow{a} [\gamma := k\ ,\ \delta := \beta + k']s'$.

If $\overline{c}(\gamma) = (\gamma, 0)$, counter γ is said to be *inactive* in the transition.
If $\overline{c}(\gamma) = (\gamma, k)$ with $k > 0$, i.e. $\gamma := \gamma + k$, counter γ is said to be *increased* in the transition.
If $\overline{c}(\gamma) = (0, k)$, i.e. $\gamma := k$ counter γ is said to be *reset* in the transition.

If $\overline{c}(\gamma) = (\delta, k)$ with $\gamma \neq \delta$; i.e. $\gamma := \delta + k$, counter γ is said to *copy* counter δ in the transition.

To access individual elements of the labels, $ltr(s \xrightarrow[\overline{c}]{a} s') = a$ and $ctr_cmd(s \xrightarrow[\overline{c}]{a} s') = \overline{c}$refer to letter and counter command of the transition.

The acceptance condition Φ is a positive Boolean combination of statements of the form

B-conditions	$\limsup \gamma < \infty$
Büchi-conditions	s appears infinitely often

for counters γ and states s.

The acceptance of a run is determined by the extended LTS
$\overline{B} = \left\langle \mathcal{S} \times \mathbb{N}^{\mathcal{C}}, \mathcal{S}_0 \times \{0\}^{\mathcal{C}}, ((\mathcal{C} \cup \{0\}) \times \mathbb{N})^{\mathcal{C}}, \overline{\mathcal{T}}, \overline{\Phi} \right\rangle$.

$\vec{s} \in \mathcal{S} \times \mathbb{N}^{\mathcal{C}}$ is called a configuration. $\vec{s}(0) \in \mathcal{S}$ gives the state of the B-automaton and $\vec{s}(\gamma)$ gives the value of counter γ. All counters have value 0 initially.
The elements of $\overline{\mathcal{T}}$ are written as $\vec{s} \xrightarrow[\overline{c}]{a} \vec{s}'$ and fulfill the following:

- $\vec{s}(0) \xrightarrow[\overline{c}]{a} \vec{s}'(0)$ is a transition $\in \mathcal{T}$
- $\vec{s}'(\gamma) = \begin{cases} \vec{s}(\delta) + k & , \overline{c}(\gamma) = (\delta, k) \\ k & , \overline{c}(\gamma) = (0, k) \end{cases}$

For a path $\vec{\pi}$ in $\overline{B}$, $\vec{\pi}(0)$ refers to the corresponding path in B and $\vec{\pi}(\gamma)$ refers to the sequence of the values of counter γ. $\overline{\Phi}$ states the conditions of Ψ formally:

Φ	$\overline{\Phi}$
$\limsup \gamma < \infty$	$\limsup \vec{\pi}(\gamma) < \infty$
s appears infinitely often	$\{i \in \mathbb{N} : src(\vec{\pi}(0)_i) = s\} = \infty$

If these conditions are met for a path $\vec{\pi}$ in $\overline{B}$ and the initial configuration $src(vec\pi)$ is in $\mathcal{S}_0 \times \{0\}^{\mathcal{C}}$, then $\vec{\pi}$ is considered a run.
For every path π of the B-automaton B there is exactly one path $\vec{\pi}$ in $\overline{B}$, with $\vec{\pi}(0) = \pi$. π is considered a run, iff $\vec{\pi}$ is a run.

For a path π, let $wrd(\pi)$ be the word created by π, i.e. the sequence $ltr(\pi_0), ltr(\pi_1), \ldots$. That word is said to be *accepted* by the run π and by the B-automaton B.

CHAPTER 3

Rendezvous-Multiple Broadcast networks

This section introduces Rendezvous-Multiple Broadcast networks, which are an extension of the Broadcast-Rendezvous networks described in [ARZS14].

These networks consist of several processes, which all run the same code. The code is represented by process templates, which are labeled transition systems. The processes have two ways of communicating with each other: rendezvous and symmetric broadcasts. Broadcasts consist of a message sent to all processes. In rendezvous transitions a fixed number K of processes communicate with each other. There are K roles for each rendezvous letter, each of those with a unique transition in the template.

All processes can communicate with any other process in the network, i.e. there is no topology between the processes involved.

3.1 Formal Definition

Both the templates and the networks of RMBNs are Labeled Transition Systems.

Definition 3. P: Rendezvous-Multiple Broadcast Network Template

The templates for the rendezvous-multiple broadcast network are an LTS of the form $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}_r \times [K] \cup \mathcal{A}_b, \mathcal{T} \rangle$. The set $\mathcal{S}$ are the states and $\mathcal{S}_0$ the initial states. $\mathcal{A}_r$ is the set of rendezvous letters, $\mathcal{A}_b$ is the set of broadcast letters and $K \in \mathbb{N}$ is the number of roles in rendezvous transitions.
The transitions $\mathcal{T}$ consist of rendezvous and broadcast transitions. The rendezvous transitions are those transitions with labels $(a, k) \in \mathcal{A}_r \cup [K]$. They are written in the

form $s \xrightarrow{a,k} s'$. In these, $s, s' \in \mathcal{S}$, $a \in A_r$ and $k \leq K$. The broadcast transitions are those with label $\beta \in \mathcal{A}_b$. They are written in the form $s \xrightarrow{\beta} s'$. For each state $s \in \mathcal{S}$ and each broadcast label $\beta \in A_b$, there is exactly one state $s' \in \mathcal{S}$ such that $s \xrightarrow{\beta} s' \in \mathcal{T}$.

src, *dst* and *label* are defined for transitions of $RMBN$ templates as they are for transitions of any LTS. Therefore $label(s \xrightarrow{a,k} s') = (a, k)$. To access a and k individually, $ltr(s \xrightarrow{a,k} s') = a$ and $role(s \xrightarrow{a,k} s') = k$ can be used.

The LTS P^N and P^∞ are constructed from the RMBN template P. They describe networks of N or any number of processes respectively.

Construction 1. P^N: Rendezvous-Multiple Broadcast Network with N processes constructed from the template P

For $N \in \mathbb{N}$ the network $P^N = \left\langle \mathcal{S}^N, \mathcal{S}_0^N, \mathcal{A}_r \times [N]^K, \mathcal{T}^N \right\rangle$ consists of N processes. The elements of $\mathcal{S}^N$ are called configurations and are N dimensional vectors over $\mathcal{S}$. For $n \leq N$ $\vec{s}(n)$ is the state process n is in when the network is in configuration $\vec{s}$. The initial configurations $\mathcal{S}_0^N$ are all configurations where every process is in an initial state. Compared to templates, the labels use K-dimensional vectors $\vec{p} \in [N]^K$ instead of roles $k \in K$. $\vec{p}(k)$ gives the process taking role K. In this definition, the number of roles in rendezvous transitions is fixed as K over all rendezvous letters. In some examples and specific networks, the number of roles can vary over rendezvous letters.
Fixing the number of roles does not reduce the expressing power of the mode. This is due to the fact that the number of roles in each global rendezvous transition can be increased to K by adding one state and a transition with source and destination in that state for every missing role.

The elements of $\mathcal{T}^N$ are called global transitions. Global rendezvous transitions are written in the form $\vec{s} \xrightarrow{a,\vec{p}}, \vec{s'}$, where $\vec{s}$ and $\vec{s'}$ are configurations, $a \in \mathcal{A}_r$ and $p \in [N]^K$. For $\vec{s} \xrightarrow{a,\vec{p}} \vec{s'}$ to be a global rendezvous transition $\vec{p}$ must be injective, i.e. $\vec{p}(k) = \vec{p}(k') \Rightarrow k = k'$. Each role is taken by a different process. Additionally,

$$\begin{aligned} &\vec{s}(n) = \vec{s'}(n) && , \forall n \notin range(f) \\ &\vec{s}(f(k)) \xrightarrow{a,k} \vec{s'}(f(k)) \in \mathcal{T} && , \forall k \leq K \end{aligned}$$

The labels are required to be *unique*, i.e. $label(t) = label(t') \Rightarrow t = t'$. Given a network without unique labels, a network with unique labels and the same runs and executions can be constructed by creating a new label for each possible selection of transitions participating in a rendezvous transition.

Process n is said to be *active* in the rendezvous transition $\vec{s} \xrightarrow{a,\vec{p}} \vec{s'}$ iff $n \in range(\vec{p})$.

Global broadcast transitions are written in the form $\vec{s} \xrightarrow{\beta} \vec{s'}$ where $\vec{s}$ and $\vec{s'}$ are configurations and $\beta \in \mathcal{A}_b$. For $\vec{s} \xrightarrow{\beta} \vec{s'}$ to be a global broadcast transition, $\vec{s}(n) \xrightarrow{\beta} \vec{s'}(n) \in \mathcal{T}$

must be true for all $n \leq N$. Every process is active in every broadcast transition.

For a global transition $\vec{t}$, $\vec{t}(n)$ refers to the transitions process n takes. For a global rendezvous transition, $\vec{t}(n) = \bot$, iff process n is not active in the transition. Additionally, $ltr(\vec{s} \xrightarrow{a,\vec{p}} \vec{s}') = a$ and $role(\vec{s} \xrightarrow{a,\vec{p}} \vec{s}') = \vec{p}$.

Figure 2.1 in Chapter 2 shows an RMBN template.

Construction 2. P^∞: Rendezvous-Multiple Broadcast Network with any number of processes constructed from template P

The complete set of networks defined by the template $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{T} \rangle$ is the LTS

$$P^\infty = \left\langle \bigcup_{N \in \mathbb{N}} \mathcal{S}^N, \bigcup_{N \in \mathbb{N}} \mathcal{S}_0^N, \mathcal{A}_r \times \mathbb{N}^K, \bigcup_{N \in \mathbb{N}} \mathcal{T}^N \right\rangle$$

, i.e. all networks for any number of processes N.

Runs and executions

The template P and the network P^∞ are both Labeled Transition Systems. Therefore, runs are already defined for both. Projections and Executions are specific to networks though.

Definition 4. Projections and Executions of RMBN

Let $\vec{\pi}$ be a global run. For process n let $i_1, i_2, \ldots,$ be the indices of those transitions in which process n is active. Then $proj(\vec{\pi}, n)$ refers to the *projection* of the global run $\vec{\pi}$ onto process n. I.e. the sequence of the local transitions process n takes in the global run $\vec{\pi}$. All global transitions where process n is not active are skipped. Formally, $proj$ is defined via the identity $proj(\vec{\pi}, n)_r = \vec{\pi}_{i_r}(n)$.
By definition, the projection of a global run onto a process is always a local run. If for a local run π, there is a global run $\vec{\pi}$ and a process n, such that $\pi = proj(\vec{\pi}, n)$, then π is called an *execution.*

Not every local run is an execution. Due to symmetry, if π is an execution, there is a global run $\vec{\pi}$ such that $proj(\vec{\pi}, 1) = \pi$.
$exec(P)$ refers to the set of all executions of the template P. $exec^{fin}(\mathcal{P})$ is the set of all finite executions and $exec^{inf}(\mathcal{P})$ the set of all infinite executions of $\mathcal{P}$. Note that the projection of an infinite run might be a finite execution. Additionally, for $\mathcal{S}_0' \subseteq \mathcal{S}_0$ the expression $exec(\mathcal{P}, \mathcal{S}_0')$ is used to denote those executions starting in any $s \in \mathcal{S}_0'$, i.e. $exec(\mathcal{P}, \mathcal{S}_0') = \{\pi \in exec(\mathcal{P}) : src(\pi) \in \mathcal{S}_0'\}$. That implies that $exec(\mathcal{P}) = exec(\mathcal{P}, \mathcal{S}_0)$.

Definition 5. PMCP: Parametrized Model Checking Problem

For a specification language Φ the finite/infinite PMCP is the following: given a template P and a specification $\phi \in \mathcal{L}$, are all executions in $exec^{fin}(\mathcal{P})$ ($exec^{inf}(\mathcal{P})$ respectively) satisfying ϕ?

For finite executions, the specification languages will be regular languages. The finite PMCP will be proven to be decidable for regular languages.

The specification used for infinite execution will be the following: Given sets $\mathcal{S}_0' \subseteq \mathcal{S}_0$ and $\mathcal{Z} \subseteq \mathcal{S}$, do all executions starting in $\mathcal{S}_0'$ reach a state in $\mathcal{Z}$? The infinite PMCP turns out to be undecidable for this kind of specifications.

3.2 Finite Executions

In order for a process to take a rendezvous transition, there have to be other processes in the sources of the other transition with the same letter. That can only happen when those states are reachable. The set of reachable states depends on the broadcast transitions the network took previously.
A breadth first search similar to [ARZS14] can be used to determine, which transitions are usable in which situation. The algorithm creates the unwinding template. The first main building block of the unwinding template is a *component.* Each component contains the states and rendezvous transitions that are reachable only via rendezvous transition from a given subset of the states. The broadcast transitions connect these components. All broadcast transitions with the same letter and source in the same component have their destination in the same component.

For only one type of broadcast letter, [ARZS14] shows that the components are arranged in a lasso structure. Here, the components can be arranged in any structure.

The second main part of calculating the unwinding template is the *reach* function. Given a component and a broadcast letter, it calculates the destination of all transitions with the broadcast letter as label and a copy of the source in the component.

Algorithm 1. Constructing the unwinding template $P^{-\circ}$ from the RMBN template P

Components are LTS characterized by their initial states $\tilde{\mathcal{S}} \subset \mathcal{S}$ and called *comp*$(\tilde{\mathcal{S}})$. The states of *comp*$(\tilde{\mathcal{S}})$ are a subset of $\mathcal{S} \times \{\tilde{\mathcal{S}}\}$, i.e. pairs of the form $(s, \tilde{\mathcal{S}})$ with $s \in \mathcal{S}$. The labels are $\mathcal{A}_r \times \{\tilde{\mathcal{S}}\} \times [K]$, i.e. triples of the form $(a, \tilde{\mathcal{S}}, k)$.

Given a set of a component's initial states $\tilde{\mathcal{S}}$, the component *comp*$(\tilde{\mathcal{S}})$ is calculated as follows:
comp$(\tilde{\mathcal{S}})$ starts the LTS $\left\langle \{(s, \tilde{\mathcal{S}}) : s \in \tilde{\mathcal{S}}\}, \{(s, \tilde{\mathcal{S}}) : s \in \tilde{\mathcal{S}}\}, \mathcal{A}_r \times \{\tilde{\mathcal{S}}\} \times [K], \emptyset \right\rangle$.

The algorithm repeatedly checks all rendezvous letters not yet added to the component. Let S_a be the set of states s for which there is a transition with letter a and source s in P. Let S'_a be the set of states s, for which there is a transition with letter a and with destination s' in P. Letter a is *added*, if $(s, \tilde{\mathcal{S}})$ is already the states of $comp(\tilde{\mathcal{S}})$ for all $s \in S_a$. If this is the case, $(s', \tilde{\mathcal{S}})$ is added to the states of $comp(\tilde{\mathcal{S}})$ for all states s' in S'_a. Additionally, for each transition $s \xrightarrow{a,k} s'$ with letter a the transition $(s, \tilde{\mathcal{S}}) \xrightarrow{a,\tilde{\mathcal{S}},k} (s', \tilde{\mathcal{S}})$ is added to transitions of $comp(\tilde{\mathcal{S}})$. From now on, a no longer needs to be checked when calculating $comp(\tilde{\mathcal{S}})$. The calculation of a component is finished, when all letters not yet added have been negatively checked since the last time a letter was added.

The second main part of the calculation of $P^{\multimap}$ is *reach*.
Let $comp(\tilde{\mathcal{S}}) = \left\langle \{(s, \tilde{\mathcal{S}}) : s \in \hat{\mathcal{S}}\}, \{(s, \tilde{\mathcal{S}}) : s \in \tilde{\mathcal{S}}\}, \mathcal{A}_r \times \{\tilde{\mathcal{S}}\} \times [K], \mathcal{T}^{\tilde{\mathcal{S}}} \right\rangle$, here, $\hat{\mathcal{S}}$ is the subset of the original states $\mathcal{S}$ corresponding to the states of $comp(\mathcal{S})$. Given a broadcast letter β, let $reach(comp(\tilde{\mathcal{S}}), \beta)$ be the set $\{s' \in \mathcal{S} \;:\; \exists s \in \hat{\mathcal{S}} : s \xrightarrow{\beta} s' \in \mathcal{T}\}$.

The unwinding template is an RMBN created by combing several components and adding broadcast transitions connecting the components. Its calculation starts with the calculation of the initial component $comp(\mathcal{S}_0)$. When the calculation of a component $comp(\tilde{\mathcal{S}})$ is finished, it is added to $P^{\multimap}$ by adding all states, labels and transitions of the component to the states, labels and transitions of $P^{\multimap}$. Then, the algorithm goes through all broadcast labels β. For every broadcast letter β, $reach(comp(\tilde{\mathcal{S}}), \beta)$ is calculated. For all broadcast transitions $s \xrightarrow{\beta} s'$ with $s \in \hat{\mathcal{S}}$, the set of states whose copies are in $comp(\tilde{\mathcal{S}})$, the transition $(s, \tilde{\mathcal{S}}) \xrightarrow{\beta} (s', reach(comp(\tilde{\mathcal{S}}), \beta)$ is added to the transitions of $P^{\multimap}$. By doing so, all states copied from $comp(\tilde{\mathcal{S}})$ have a broadcast transition labeled with β starting in it. Additionally, if $comp(reach(comp(\tilde{\mathcal{S}}), \beta)$ has not yet been calculated and it is not yet in the queue, $reach(comp(\tilde{\mathcal{S}}), \beta)$ is added to the queue of components that have to be calculated. When *reach* has been calculated for all broadcast letters, the first element of the queue are the initial states of the next component to be calculated.

This queue starts with only the set of the original initial states $\mathcal{S}_0$ as the initial states for the first component to be calculated. The calculation then iterates between calculating *comp* and *reach*. When the queue is empty after *reach* has been calculated for every broadcast letter for the last component, the entire calculation is finished. As the initial states of each component is the subset of all states, at most $2^{|\mathcal{S}|}$ components have to be calculated, i.e. the number of components is at worst exponential in the number of states.
Only the initial states of the first component $comp(\mathcal{S}_0)$ are the initial states of the unwinding template $P^{\multimap}$.

Figure 3.1 shows the unwinding of the network in Figure 2.1. There never can be processes in all three of the states A, B and C. Depending on the sequence of previous broadcasts, only two of them are reachable. Therefore, there are three components with

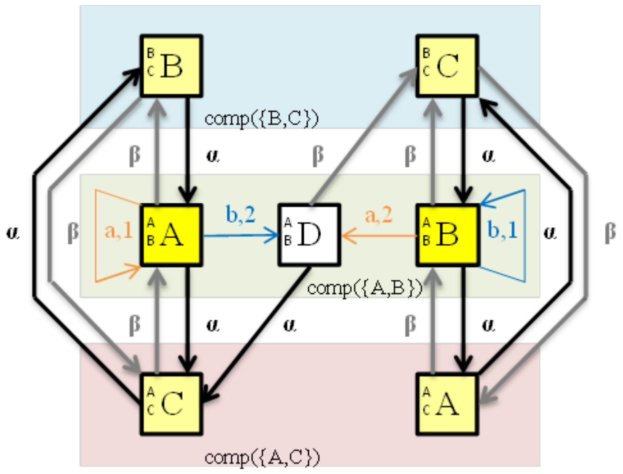

Figure 3.1: Unwinding template of the network in Figure 2.1

initial states $\{A, B\}$, $\{A, C\}$ and $\{B, C\}$ respectively. In Figure 3.1 every component has its own background color. As long as a process does not take any rendezvous transitions, it stays either on the left hand side or the right hand side of Figure 3.1. The rendezvous transitions give processes the opportunity to change side, as long as there is a process on the other side.

Definition 6. Attributing components to transitions and runs

If $t : s \xrightarrow{a,k} s'$ is a rendezvous transition in P let $(t, \tilde{\mathcal{S}})^{\multimap}$ be the transition $(s, \tilde{\mathcal{S}}) \xrightarrow{(a,\tilde{\mathcal{S}},k)} (s', \tilde{\mathcal{S}})$. For a broadcast transition $t : s \xrightarrow{\beta} s'$, $(t, \tilde{\mathcal{S}})^{\multimap}$ is the transition $(s, \tilde{\mathcal{S}}) \xrightarrow{\beta} (s', reach(comp(\tilde{\mathcal{S}}), \beta)$. A local or global run in P can be transformed into a local or global run in $P^{\multimap}$. For a local run π, let $cmpi(\pi, i)$ be the initial states of the component the corresponding run of $P^{\multimap}$ is in after the i-th transition. It is iteratively defined: $cmpi(\pi, i) =$

$$= \begin{cases} \mathcal{S}_0 & , \; i = 0 \\ cmpi(\pi, i-1) & , \; \pi_i \text{ is a rendezvous transition} \\ reach(comp(cmpi(\pi, i-1)), \beta) & , \; \pi_i \text{ is a broadcast transition labeled with } \beta \end{cases}$$

The sequence of transitions $\pi^{\multimap}$ is now defined as $\pi_i^{\multimap} = (\pi_i, cmpi(\pi, i-1))^{\multimap}$. $dst(\pi_{i-1}^{\multimap}) = src(\pi_i^{\multimap})$ holds true, because the component of the destination in $t^{\multimap}$ is consistent with the definition of $cmpi(\pi, i)$ for the run π.

For a configuration $\vec{s}$ let $(\vec{s}, \tilde{\mathcal{S}})^{\multimap}$ be the configuration of states with $(\vec{s}, \tilde{\mathcal{S}})_n^{\multimap} = (\vec{s}_n, \tilde{\mathcal{S}})$ for every process n. Using this, the definition of $^{\multimap}$ can be naturally extended to global transitions $\vec{t}$ and global runs $\vec{\pi}$.

If $\vec{\pi}$ is a global run of $\mathcal{P}$ then $\vec{\pi}^{-\circ}$ is a global run of $\mathcal{P}^{-\circ}$.
If π is a local run of P then $\pi^{-\circ}$ may or may not be a local run of $P^{-\circ}$, as the required states or transitions may not be part of a component. The runs of $P^{-\circ}$ correspond to the executions of P. To proof this, a couple of definitions and lemmata are used:
Let τ be a sequence of broadcast letters. Let τ_i be the i-th label in this sequence. Then,
$$cmpi(\tau, i) = \begin{cases} \mathcal{S}_0 & , i = 0 \\ reach(comp(cmpi(\tau, q-1)), \tau_i) & , i \geq 1 \end{cases}.$$

The same definition for *broad* is applied to global runs $\vec{\pi}$. For a local run π, let $dstcmpi(\pi) = cmpi(\pi, |\pi|)$, i.e. the component at the end of the run. The same definition for *dstcmpi* is used for global runs $\vec{\pi}$ and sequences of broadcast labels τ.

Let $^{\circledcirc}$ be the inverse function of $^{-\circ}$, i.e. it removes the components from states, configurations, transitions and runs.
For runs of the unwinding $P^{-\circ}$ it is true that $(\pi^{\circledcirc})^{-\circ} = \pi$ and $(\vec{\pi}^{\circledcirc})^{-\circ} = \vec{\pi}$. This is due to all transitions in $P^{-\circ}$ being consistent with the definition of *cmpi*.

The following lemma states the consistency of Definition 6 and follows immediatly from it:

Lemma 1. *For a local or global run π or $\vec{\pi}$, let $broad(\pi)$ be the sequence of all broadcast letters appearing in π and let i_r be the sequence of the corresponding indices. It holds true that $cmpi(broad(\pi), r) = cmpi(\pi, i)$ for all i with $i_r \leq i < i_{r+1}$*

The following lemma allows merging global runs, as long as the have the same broadcast transitions:

Lemma 2. *Let ${}_1\vec{\pi}, {}_2\vec{\pi}, \ldots, {}_M\vec{\pi}$ be a family of M global runs with $broad({}_m\vec{\pi}) = broad({}_{m'}\vec{\pi})$ for all m, m'.*
Then, there is a merged global run $\vec{\pi}$ with $dim(\vec{\pi}) = \sum_{m=1}^{M} dim({}_m\vec{\pi})$.
It has $proj\,(\vec{\pi}, \sum_{\tilde{m}=1}^{m} dim({}_m\vec{\pi}) + n) = proj({}_m\vec{\pi}, n)$ for all $n \leq dim({}_m\vec{\pi})$.

Proof: $\vec{\pi}$ can be constructed by simply putting all processes from each run ${}_m\vec{\pi}$ next to each other. First, it takes all rendezvous transitions from the first run ${}_1\vec{\pi}$ before the first broadcast transition. Then all rendezvous transitions before the first broadcast transition in the second run. It continues that way until all rendezvous transitions from the M-th run before the first broadcast transitions are done. Then, it takes the first broadcast transition, which must have the same letter in all M runs. This puts all processes in the same state they have after the first broadcast transitions in their respective run. $\vec{\pi}$ then does all rendezvous transitions between the first and second rendezvous transition and so forth. This process leads to a run whose projections are the same as the projections of the original run. □

The following lemma shows that all states in $P^{-\circ}$ are reachable:

Lemma 3. *Let P be a rendezvous-multiple broadcast template and $P^{\multimapinv}$ its unwinding template. Given a state $(s, \mathcal{S}')$ in component $comp(\mathcal{S}')$ of the unwinding template $P^{\multimapinv}$ and sequence of broadcast labels τ with $dstcomp(\tau) = \mathcal{S}'$. Then, there exists a global run of $\mathcal{P}^{\multimapinv}$ $\vec{\pi}$ with Process 1 in state $(s, \mathcal{S}')$ at the end of the run and $broad(\vec{\pi}) = \tau$.*

Proof via nested induction: The outer induction is over the sequence of broadcast labels τ, the inner is over the states of $comp(\mathcal{S}')$ in the order they were added to $comp(\mathcal{S}')$.

The outer induction hypothesis states that Lemma 3 is true for all prefixes τ' of τ and all states $(\hat{s}, dstcomp(\tau'))$ in the component τ' leads to.

The inner induction hypothesis states that Lemma 3 is true for all states that were added to $comp(\mathcal{S}')$ before s.

First, lets assume $s \in \mathcal{S}'$, i.e. s is an initial state of its component. If $length(\tau) = 0$, then $\mathcal{S}' = \mathcal{S}_0$. In this case, the global run of length 0 only consisting of one process in state s is used for $\vec{\pi}$.

Otherwise, there is a state corresponding to $\hat{s}$ in the component $comp(\tau, |\tau| - 1)$ and $\hat{s} \xrightarrow{\tau_{|\tau|}} s$ is a broadcast transition in P. Using the outer induction hypothesis there is a global run π' with broadcasts $\tau_1, \ldots, \tau_{|\tau|-1}$. At the end of π' Process 1 is in state $(\hat{s}, comp(\tau, |\tau| - 1))$. Adding a broadcast transition labeled with $\tau_{|\tau|}$ yields the desired global run π.

For the case of $s \notin \mathcal{S}'$ let a be the letter used to add $(s, \mathcal{S}')$ to $comp(\mathcal{S}')$. Then there is a state $\hat{s}$ and a role k', such that $\hat{s} \xrightarrow{a,k'} s$ is a rendezvous transition in P. All states in S_a, the set of starting states of transitions labeled with a, have been added to $comp(\mathcal{S}')$ before s. Let ${}_k\hat{s}$ be the starting state of the transition labeled with a and identifier k. Using the inner induction hypothesis, there are K runs ${}_k\vec{\pi}$. For them, $broad({}_k\vec{\pi}) = \tau$ and the destination of Process 1 is ${}_k\hat{s}$.
Lemma 2 allows merging the runs ${}_k\vec{\pi}$, using ${}_{k'}\vec{\pi}$ as first run. Now, a rendezvous transition with label a using the first process of each run can be added. This yields a run where Process 1 has destination s. □

Lemma 3 together with Lemma 2 allows loading as many processes as needed into any state of a component of $P^{\multimapinv}$.

Now, the final theorem for finite executions can be formulated:

Theorem 1. *Let P be a rendezvous-multiple broadcast network template and $P^{\multimapinv}$ its unwinding template. If π is a finite run of P, then π is an execution of P iff $\pi^{\multimapinv}$ is a finite run of $P^{\multimapinv}$.*
Every finite run of $P^{\multimapinv}$ is an execution.

Proof: Let us show first that every run π of $P^{-\circ}$ is an execution. This is done iteratively over the transitions of π. Let $\vec{\pi}(\pi, i)$ be the global run allowing the first i transitions of π. $\vec{\pi}(\pi, 0)$ is the configuration containing only one process in state $src(\pi)$. If π_i is a rendezvous transition and $label(\pi_i) = a$ then a copy of all states in S_a, the set of sources of transitions with letter a, must be present in $comp(cmpi((\pi, i-1))$. For all of these states $(s, cmpi(\pi, i-1))$, Lemma 3 yields a run with a process in state $(s, cmpi(\pi, q))$ at the end and broadcast transitions $broad(\pi_1, \ldots, \pi_i)$. These runs are merged with $\vec{\pi}(\pi, i-1)$ using Lemma 3. Then, a transition with label a can be added using the first process of each of these runs.
If $trans(\pi, 1)$ is a broadcast transition with label β, the global broadcast transition with label β is simply added to $\vec{\pi}(\pi, i-1)$.
The global run constructed this way allows π.

Second, it is shown that π is an execution of P iff $\pi^{-\circ}$ is a run of $P^{-\circ}$.
$\Rightarrow$: If π is an execution, there is a global run $\vec{\pi}$ with $proj(\vec{\pi}, 1) = \pi$. Then $proj(\vec{\pi}^{-\circ}, 1) = \pi^{-\circ}$. It remains to show that $\vec{\pi}^{-\circ}$ is a global run, i.e. that each transition in $\vec{\pi}^{-\circ}$ is in $P^{-\circ}$. Again, induction over the transitions of $\vec{\pi}^{-\circ}$ is used. If all transitions up to the i-th transition are in $P^{-\circ}$, then all states in the configuration $dst(\vec{\pi}_i^{-\circ})$ are represented component $comp(cmpi(\vec{\pi}^{-\circ}, i))$. If $\vec{\pi}_i^{-\circ}$ is a rendezvous transition, all states required for it are therefore present in $comp(cmpi(\vec{\pi}^{-\circ}, i)$. This implies that $\vec{\pi}_i^{-\circ}$ is also present for that component. Broadcast transitions can always be taken, as there is a broadcast transition starting in every state.
$\Leftarrow$: If $\pi^{-\circ}$ is a run in $P^{-\circ}$, it is an execution. Therefore, there is a global run $\vec{\pi}$ with $proj(\vec{\pi}, 1) = \pi^{-\circ}$. Then, $proj(\vec{\pi}^{\circledcirc}, 1) = \pi$. □

Therefore, the question whether a finite run is an execution is decidable. This is unfortunately not true for infinite executions.

3.3 Infinite executions

In networks infinite runs are way harder to analyze than finite runs. The most straightforward idea would be to interpret $P^{-\circ}$ as a Büchi-automaton, but not every run of that Büchi-automaton is an execution.

To see that, let us look at Figure 3.1, which shows the unwinding template of the network in Figure 2.1. The unwinding shows that without rendezvous transitions, all processes stay either in the left or the right side of the graphic. Processes can only change from right to left when the network takes an β-broadcast out of component $comp(\{A, B\})$. They can only change from right to left, when the network takes an α-broadcast out of the same component. Whenever a process takes the transition labeled with $(a, 1)$, another process moves to D. If the next broadcast is an α-broadcast, then that other process has moved from the right hand side states to the left hand side states. As there is only a finite number of processes, this can only a finite number of times, unless processes come

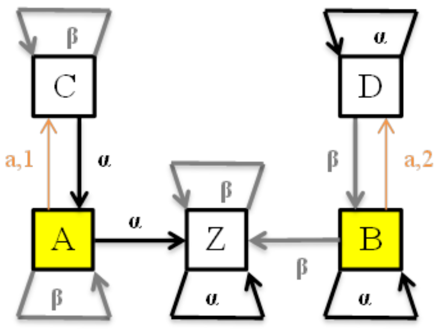

Figure 3.2: Example of a network whose executions are not ωB-regular

back. This can only happen when the the network takes an β-broadcast out of component $comp(\{A, B\})$. Therefore, the number of times a process can take the transition labeled with $(a, 1)$ between the times the network leaves component $comp(\{A, B\})$ using a β-broadcast is bounded by the number of processes in the network. As a consequence, only runs where the number of $a, 1$ labeled transitions between β-broadcast leaving component $comp(\{A, B\})$ is bounded are executions. The same is true for $b, 1$ and α-broadcasts leaving component $comp(\{A, B\})$.

In [ARZS14], the PMCP is shown to be decidable for one type of broadcast message. They search for cycles containing any given transition. In their case, they could transform any cycle with broadcasts into a cycle with a fixed number of broadcasts. Therefore, they only had to search for cycles with that many broadcasts. The example in Figure 3.1 shows that this approach can not be used here easily, as there are many potential cycles that do or do not have group broadcasts leaving component $comp(\{A, B\})$.

[ARZS14] showed that the language created by the executions of rendezvous-broadcast networks are always ωB-regular, as defined in Definition 2. The language of the network in Figure 2.1 is still ωB-regular. It can be shown though that he executions of the network in Figure 3.2 are not ωB-regular though.

Simulating counter machines with multiple broadcast networks

This section will show that the PMCP for infinite runs is undecidable for rendezvous-multiple broadcast networks. This result is analogous to the case of asymmetric broadcast [EFM99], where it is shown that two types of asymmetric broadcasts are enough for undecidability. The proof here is based on their proof. It is shown that networks can simulate counter machines. Counter machines consist of a process moving from state to state and are equipped with some counters [Min67]

Definition 7. Counter machines

Given a set of counters $\mathcal{C}$, the template for counter machines are the labeled transition systems $\left\langle \mathcal{S}, \{{}_0s\}, \overline{\mathcal{C}}, \mathcal{T} \right\rangle$, where $\overline{\mathcal{C}}$ is a set of counter commands. It it the union $\bigcup_{c \in \mathcal{C}} \{c := c+1, c := c-1, c = 0\}$. There are therefore three types of transitions:

$s \xrightarrow{c:=c+1} s'$	Increases counter c.
$s \xrightarrow{c:=c-1} s'$	Decreases counter c. Can only be taken when $c > 0$
$s \xrightarrow{c=0} s'$	Zero-Test. Can only be taken when $c = 0$

Configurations are elements of $\mathcal{S} \times \mathbb{N}^{\mathcal{C}}$. For a configuration $\vec{s}$ and a counter $c \in \mathcal{C}$, $\vec{s}(c)$ refers to the value of counter c in the configuration $\vec{s}$. $\vec{s}(0) \in \mathcal{S}$ refers to the *control* of the counter machine. $\mathcal{T}^{C} \subset \mathcal{S} \times \mathbb{N}^{\mathcal{C}} \times \overline{\mathcal{C}} \times \mathcal{S} \times \mathbb{N}^{\mathcal{C}}$ is the set of global transitions. Global transitions are written in the form $\vec{s} \xrightarrow{\overline{c}} \vec{s'}$, where $\vec{s}$ and $\vec{s'}$ are configurations and $\overline{c} \in \overline{\mathcal{C}}$ is a counter command. $\vec{s} \xrightarrow{\overline{c}} \vec{s'}$ is a global transition, if

- $\vec{s}(0) \xrightarrow{\overline{c}} \vec{s'}(0)$ is a transition $\in \mathcal{T}$
- $\vec{s'}(\gamma) = \begin{cases} \vec{s}(\gamma) + 1 & , \overline{c} \ = \ (\gamma := \gamma + 1) \\ \vec{s}(\gamma) - 1 & , \overline{c} \ = \ (\gamma := \gamma - 1) \\ \vec{s}(\gamma) & , \text{otherwise} \end{cases}$
- If $\overline{c} \ = \ (\gamma = 0)$ then $\vec{s}(\gamma)$ must be 0.

The otherwise case occurs when $\overline{c} \ = \ (\gamma = 0)$ or when $\overline{c}$ involves another counter than γ.
A counter machine is therefore the LTS $\left\langle \mathcal{S} \times \mathbb{N}^{\mathcal{C}}, \{({}_0s, 0, \ldots, 0)\}, \overline{\mathcal{C}}, \mathcal{T}^{C} \right\rangle$.

If $\vec{\pi}$ is a path of a counter machine, $src(\vec{\pi}_i)(\gamma)$ refers to the value of counter γ before the i-th transition.
An infinite path $\vec{\pi}$ of M is called bounded, if $src(\vec{\pi}_i)(\gamma) < V$ for some $V \in \mathbb{N}$, all $c \in \mathcal{C}$ and all $i \in \mathbb{N}$.

The main challenge compared to [EFM99] is the lack of a controller. They can use the controller process to simulate the control of the counter machine, while all other processes of the networks simulate the counters. Networks with symmetric broadcasts have the property that if one process can behave according to a given execution, any number of process can follow the same execution. Therefore, several processes can take the role of the control of the counter machine. This problem is overcome by using broadcast transitions to force all processes simulating the control to have the same transitions.
In [EFM99] the number of processes in a given state simulates the value of counters. Here, these values are represented by that number of processes in a designated state

divided by the number of processes simulating the control of the counter machine.

They prove in chapter five that the following question is undecidable: Does a given network with rendezvous transitions and asymmetric broadcasts allow infinite runs starting in a given state?
We reduce infinite PMCP for RMBN to the same undecidable question about counter machines [EFM99],[Min67]:

Problem 1. *Let M be a counter machine with 3 counters, where every infinite bounded run (not necessarily initial) visits the initial state infinitely often. Is there any infinite bounded run of M?*

Construction 3. RMBN template P_M from the counter machine M

To simulate a counter machine $M = \left\langle \mathcal{S}, \{{}_0s\}, \overline{\mathcal{C}}, \mathcal{T} \right\rangle$, we create a rendezvous-broadcast network with $|\mathcal{T}|$ broadcast transitions: $P_M = \left\langle \tilde{\mathcal{S}}, \tilde{\mathcal{S}}_0, \tilde{\mathcal{A}}_r \times [2] \cup \tilde{\mathcal{A}}_r, \tilde{\mathcal{T}} \right\rangle$.

- The set of states $\tilde{\mathcal{S}}$ are $\{A, Z\} \cup \mathcal{C} \cup \mathcal{S} \cup \mathcal{T}$.
- The initial states $\tilde{\mathcal{S}}_0$ are $\{{}_0s, A\}$.
- The rendezvous letters $\tilde{\mathcal{A}}_r$ are a_t for $t \in \mathcal{T}$.
- The broadcast letters $\tilde{\mathcal{A}}_b$ are β_t for $t \in \mathcal{T}$.
- The transitions $\tilde{\mathcal{T}}$ are constructed from the transitions $\mathcal{T}$ as follows. All broadcast transitions missing in the table below lead to the sink Z:

Counter Machine	Rendezvous-Multiple Broadcast Network		
$t = (s \xrightarrow{\gamma:=\gamma+1} s')$	$s \xrightarrow{a_t,1} t$	$A \xrightarrow{a_t,2} \gamma$	
	$t \xrightarrow{\beta_t} s'$	$A \xrightarrow{\beta_t} A$	$\tilde{\gamma} \xrightarrow{\beta_t} \tilde{\gamma}\ ,\ \forall \tilde{\gamma} \in \mathcal{C}$
$t = (s \xrightarrow{\gamma:=\gamma-1} s')$	$s \xrightarrow{a_t,1} t$	$\gamma \xrightarrow{a_t,2} A$	
	$(B, t) \xrightarrow{\beta_t} (A, s')$	$A \xrightarrow{\beta_t} A$	$\tilde{\gamma} \xrightarrow{\beta_t} \tilde{\gamma}\ ,\ \forall \tilde{\gamma} \in \gamma$
$t = (s \xrightarrow{\gamma=0} s')$	$s \xrightarrow{\beta_t} s'$	$A \xrightarrow{\beta_t} A$	$\tilde{\gamma} \xrightarrow{\beta_t} \tilde{\gamma}\ ,\ \forall \tilde{\gamma} \neq \gamma$

The state A is acting as the zero position for all counters and Z is a sink. Processes simulating the counters can move from A to γ. The number of processes in the state γ in a run of the new network indicates the value of counter γ in the corresponding run of the counter machine. Processes in states $s \in \mathcal{S}$ and $t \in \mathcal{T}$ simulate the control of the counter machine. They guess the next transition of the counter machine t by moving to state t.

P_M is constructed such that if a process is not where it is supposed to be, it sinks with the next broadcast. The transitions of the simulated counter machine are determined by the broadcast transitions. The processes simulating the state have to guess the next transitions correctly. If they fail to do so, they sink.

A counter process sinks when it is in a state $\gamma \in \mathcal{C}$ when there is a broadcast transition b_t, where the original transition t has the counter command $\gamma = 0$. This idea was used in [EFM99] too. The whole system guesses that the counter is 0. Otherwise processes sink, which can only happen a finite number of times.

After each broadcast, no process will be in any state $\in \mathcal{T}$. Such a configuration stands for the following value of the counters γ in the original counter machine:

$$\frac{S_\gamma}{\sum_{s \in \mathcal{S}} S_s} \tag{3.3.1}$$

where $S_{\tilde{s}}$ is the number of processes in state $\tilde{s} \in \tilde{\mathcal{S}}$ in such a configuration. In every reachable configuration of the network, all processes in any state $\in \mathcal{S}$ are in the same state. This is due to the following: For every broadcast letter β_t, only one state appears as destination of broadcast transitions labeled with β_t: $dst(t)$. Therefore there is only one state $\in \mathcal{S}$ in the initial states of the component of the unwinding from Definition 6.

The following lemma contains the main part of the proof of the undecidability result in Theorem 2.

Lemma 4. *Let $M = \left\langle \mathcal{S}, \{{}_0s\}, \overline{\mathcal{C}}, \mathcal{T} \right\rangle$ be a counter machine and $\mathcal{P}_M$ be the RMBN template constructed in Definition 3. Then, M contains an infinite bounded path iff there is a projection of a global path of $\mathcal{P}_M$ to one process that visits states in $\mathcal{S}$ infinitely many times.*

Observe that a path in the template P_M visits a state in $\mathcal{S}$ infinitely many times iff it visits a state in $\mathcal{S}$ once and never reaches Z, due the fact that the only alternative to moving to Z is to alternate between states in $\mathcal{S}$ and $\mathcal{T}$.

Proof: $\Leftarrow$:

Sinking of processes can only occur a finite number of times, as it is impossible for processes to come back from Z. When no process sinks anymore, all remaining processes in states $\mathcal{S}$ and $\mathcal{T}$ have the same behavior. They follow a possible run of the counter machine, given by the broadcast transitions. Furthermore the number of processes in each counter is increased or decreased exactly by the number of processes in states in $\mathcal{S}$ between broadcasts. Therefore there exists an infinite path of M which is bounded by (3.3.1).

$\Rightarrow$:

Let π be an infinite path of M bounded by $V \in \mathbb{N}$. Then a run of $\mathcal{P}$, $\tilde{\pi}$ is constructed. The starting configuration $src(\tilde{\pi})$ has one process in $src(\pi)$, $src(\pi)(\gamma)$ processes in γ for each counter $\gamma \in \mathcal{C}$ and $V \cdot |\mathcal{C}|$ processes in A. This system is now able to correctly simulate the run of M without any process ever sinking. The single process simulating the state starting in $src(\pi)$ has the same behavior as the state of the counter machine. □

Note, that Lemma 4 is only concerned with paths for both counter machines and RMBN. They do not need to start in initial configuration. Fortunately, the counter machines in Problem 1 have the property that every infinite bounded run visits initial states infinitely many times.

Theorem 2. *The following PMCP is undecidable: Given a RMBN template $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}_r \times [K] \cup \mathcal{A}_b, \mathcal{T} \rangle$, a set of initial states $\mathcal{S}_0' \subseteq \mathcal{S}_0$ and a set of undesired states $\mathcal{Z} \subset \mathcal{S}$, do all infinite executions starting in $\mathcal{S}_0$ reach a state in $\mathcal{Z}$?*

Let M be a counter machine with 3 counters, where every infinite bounded run visits the initial state infinitely many times. If we could answer, whether every infinite run of the RMBN P_M from Construction 3 starting in the initial state of the counter machine reaches state Z, then we would know whether M contains an infinite bounded path due to Lemma 4. As this path visits the initial state infinitely many times, it contains a bounded run. This is a contradiction to the undecidability of Problem 1.

Simulating any multiple broadcast network with two broadcast types

The undecidable result in Theorem 2 only proves undecidability of the PMCP for infinite runs in general. [ARZS14] gave an algorithm to solve the PMCP for one broadcast letter. Therefore, there might be other special cases of RMBN that are decidable. The construction of the rendezvous-multiple broadcast network has as many broadcast types as the original counter machine had transitions. Therefore Theorem 2 only proves that there is no algorithm capable to solve the PMCP for networks with any number of broadcasts. Given all results until now, it would still be possible for an algorithm to exist capable of solving the PMCP for a limited number of broadcast types larger than one. Unfortunately, the PMCP for infinite runs is already undecidable for two broadcast types alone. This section will prove this by simulating any rendezvous-multiple broadcast network with another network with only two broadcast types. This is achieved by using sequences of the two new broadcast letters to encode the original broadcasts.

Construction 4. RMBN $\tilde{P}$ with two broadcast letters from RMBN P with any number of broadcast letters

Let $P = \langle \mathcal{S}, \mathcal{S}_o, \mathcal{A}_r \times [K] \cup \{\beta_1, \dots, \beta_I\}, \mathcal{T} \rangle$ be a RMBN template with I broadcast letters.
Then $\tilde{P}$ is the template $\left\langle \mathcal{S} \times \{0, \dots, I\} \cup \{Z\}, \mathcal{S}_0 \times \{0\}, \mathcal{A}_r \times [K] \cup \{\alpha, \beta\}, \tilde{\mathcal{T}} \right\rangle$. The new rendezvous transitions are $((s,0) \xrightarrow{a,k} (s',0))$ for every rendezvous transition $(s \xrightarrow{a,k} s') \in \mathcal{T}$. There are no rendezvous transitions for states with $i \neq 0$.
The broadcast transitions are:

Broadcast transition	for
$(s,i) \xrightarrow{\beta} (s,i+1)$	$s \in \mathcal{S},\ 1 \leq i < I$
$(s,I) \xrightarrow{\beta} Z$	$s \in \mathcal{S}$
$(s,i) \xrightarrow{\alpha} (s',0))$	$(s, \beta_i, s') \in {}^b\mathcal{T},\ 1 \leq i < I$
$(s,0) \xrightarrow{\alpha} Z$	$s \in \mathcal{S}$

Let h be the string homomorphism replacing each broadcast transition labeled with β_i with the corresponding sequence of broadcasts transitions labeled with $\beta, \cdots, \beta, \alpha$ consisting of i instances of β-broadcasts followed by one α-broadcast.
The state Z is called a sink.

Lemma 5. *Let P be a RMBN template and $\tilde{P}$ be the template constructed in Definition 4. Then, a run π of P is an execution iff $h(\pi)$ is an execution of $\tilde{P}$. For every execution π' of $\tilde{P}$ that does not sink there is an execution π with $h(\pi) = \pi$.*

Proof: $\Rightarrow$:
Each global run $\vec{\pi}$ of P can be directly translated into a global run of $\tilde{P}$ by extending h to apply to global transitions. This global run is witness to the fact that $h(\pi)$ is an execution.
$\Leftarrow$:
Let $\vec{\pi}'$ be a global run of $\tilde{P}^{\infty}$ that is witness that π' is an execution. A global run in P^{∞} corresponding to $\vec{\pi}'$ can be constructed: Processes that do not sink behave like a run of the original template P. For processes that do sink a corresponding finite original run can be constructed up to the transition in which it sinks. This run can be extended by the process not taking any rendezvous transition, but still participating in all broadcast transitions. These runs can be merged to form a global run in P^{∞}, which is witness that π is an execution. □

Using Lemma 5, Theorem 2 can be strengthened in the sense that the PMCP for safety specifications is already undecidable for infinite runs of rendezvous-multiple broadcast networks with two types of broadcast transitions. To be precise, the following is undecidable:

Theorem 3. *The following PMCP is undecidable: Given a RMBN template $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}_r \times [K] \cup \{\alpha, \beta\}, \mathcal{T} \rangle$ with two broadcast letters, a set of initial states $\mathcal{S}'_0 \subseteq \mathcal{S}_0$ and a set of undesired states $\mathcal{Z} \subset \mathcal{S}$, do all infinite executions starting in $\mathcal{S}_0$ reach a state in $\mathcal{Z}$?*

CHAPTER 4

Rendezvous-Fixed Group Broadcast Networks

In the last section the undecidability of the PMCP for liveliness specifications of rendezvous-multiple broadcast networks was proven. As that problem is decidable for only one broadcast type there might be other decidable special cases. The next three sections will deal with one area of special cases: Networks, where the processes are partitioned into groups. Each broadcast type only affects the processes of one group. Processes can not communicate with processes of other groups via broadcasts. Some rendezvous transitions on the other hand do allow communication between groups.
For the networks in this section the number of groups is determined by the process templates. In the networks in the next two chapters the number of groups is free and can be different for each global run of the network. Unfortunately, the PMCP for infinite runs is still undecidable for networks with a fixed number of groups. The next two chapters will show that the problem is decidable when the number of groups is not predetermined.

In rendezvous fixed-group-broadcast networks (RFGBN) process templates define the groups. Each group has its own states and transitions. This implies that the number of groups is fixed by the template.

4.1 Formal definition

The RFGBN and the templates they are defined with are LTS.

Definition 8. P: Rendezvous-Fixed Group Broadcast Network Template

The template for Rendezvous-fixed group broadcast networks (RFGBN) with L groups is the LTS $P = \langle \mathcal{S} \times [L], \mathcal{S}_0, \mathcal{A} \times [K] \cup \{\beta\}, \mathcal{T} \rangle$.
$[L]$ is a set of groups. The set of initial states $\mathcal{S}_0 \subset \mathcal{S} \times [L]$ has the property that for every

group $l \leq L$ there is at least one $s \in \mathcal{S}$ with $(s, l) \in \mathcal{S}_0$. For every pair $(s, l) \in \mathcal{S} \times [L]$ there is a group broadcast transition with source (s, l), which is most of the time written in the form $(s, l) \xrightarrow{\beta} (s', l)$. The rendezvous transitions are usually written as $(s, l) \xrightarrow{a,k} (s', l)$, where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $k \leq K$ and $l \leq L$. For both broadcast and rendezvous transitions, the group is the same in the source and the destination.

As for RMBN, $ltr(t) = a$ and $role(t) = k$ for the rendezvous transition $t = (s, g) \xrightarrow{a,k} (s', g)$.

As RFGBN are a special case of RMBN, the LTS P^N and P^∞ are constructed from the RMBN template P, comparable to Constructions 1 and 2.

Construction 5. P^N: Rendezvous-Multiple Broadcast Network with N processes constructed from the template P

For $N \in \mathbb{N}$, the network P^N is the LTS $\left\langle \mathcal{S}^N \times [L]^N, \mathcal{S}_0^N, \mathcal{A} \times [N]^K \cup \{\beta\} \times [L], \mathcal{T}^N) \right\rangle$. The pair $(\vec{s}, \vec{l}) \in \mathcal{S}^N \times \mathcal{G}^N$ is called a configuration. The initial configurations $\mathcal{S}_0^N$ is the set of configurations $(\vec{s}, \vec{l})$ with the property that $(\vec{s}(n), \vec{l}(n)) \in \mathcal{S}_0$ for all $n \leq N$.

Global rendezvous transitions $\vec{t} \in \mathcal{T}^N$ are global transitions with labels $(a, \vec{p}) \in \mathcal{A} \times [N]^K$. They are written in the form $(\vec{s}, \vec{l}) \xrightarrow{a,\vec{p}} (\vec{s'}, \vec{l})$. Note that for all rendezvous transitions the group vector $\vec{l}$ is the same for source and destination. As for all RMBN p must be injective. The following must be true:

$$\begin{array}{ll} \vec{s}(n) = \vec{s'}(n) & , \forall n \notin range(p) \\ (\vec{s}(\vec{p}(k)), \vec{l}(\vec{p}(k))) \xrightarrow{a,k} (\vec{s'}(\vec{p}(k)), \vec{l}(\vec{p}(k))) \in \mathcal{T} & , \forall k \leq K \end{array}$$

Global group broadcast transitions are those global transitions with labels (β, l) where $l \in [L]$. They are written in the form $(\vec{s}, \vec{l}) \xrightarrow{\beta,g} (\vec{s'}, \vec{l})$, where again the group vector is not changing. $(\vec{s}, \vec{l}) \xrightarrow{\beta,l} (\vec{s'}, \vec{l})$ is a global group broadcast transition iff:

$$\begin{array}{ll} \vec{s}(n) = \vec{s'}(n) & , \vec{l}(n) \neq l \\ (\vec{s}(n), \vec{l}(n)) \xrightarrow{\beta} (\vec{s'}(n), \vec{l}(n)) & , \vec{l}(n) = l \end{array}$$

Process n is said to be *active* in the group broadcast transition $(\vec{s}, \vec{l}) \xrightarrow{\beta,l} (\vec{s'}, \vec{l})$ iff $\vec{l}(n) = l$.

For a global rendezvous transition $t = (\vec{s}, \vec{l}) \xrightarrow{a,p} (\vec{s'}, \vec{l})$, $ltr(t) = a$ and $role(t) = \vec{p}$. For a global group broadcast transition $(\vec{s}, \vec{l}) \xrightarrow{\beta,l} (\vec{s'}, \vec{l})$, $ltr(t) = \beta$ and $role(t) = l$.

For every local run π, the group l stays constant in all states throughout the run. It is called the group of the run π.

Runs and executions are defined for RFGBN as they are defined for RMBN in Definition 4. Note that while in RMBN all processes are active in all broadcast transitions, in RFGBN process are only active in group broadcast transitions of their own group.

4.2 Parametrized Model Checking Problem

The RFGBN is a special case of the RMBN. Therefore, the decidability of the PMCP for finite runs follows from the decidability of the RMBN.

For infinite runs, the undecidability result can unfortunately be extended to RFGBN as well. This is done by constructing an RFGBN with two groups that simulates a RMBN with two broadcast letters.

Construction 6. RFGBN P^f from the RMBN P with two broadcast letters

Let $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}_r \times [K] \cup \{\alpha, \beta\}, \mathcal{T} \rangle$ be a RMBN template with two broadcast letters. Then, the RFGBN template P^f with rendezvous transitions of different sizes is defined as the LTS $\left\langle \mathcal{S}^f \times \{\alpha, \beta\}, \mathcal{S}_0^f, \tilde{\mathcal{A}}^f \cup \{\beta\}, \mathcal{T}^f \right\rangle$, where:

- The states in $\mathcal{S}^f \times \{\alpha, \beta\}$ are triples $((s, d), b) \in (\mathcal{S} \times \{A, B, \alpha, \beta\}) \times \{\alpha, \beta\}$, usually written in the form $s, d; b$. In addition, there are the sinks (Z, α) and (Z, β). Here, $\{\alpha, \beta\}$ is used as the set of groups instead of [2].
- The initial states $\mathcal{S}_0^f$ are the states $\{(s, A; b) : s \in \mathcal{S}_0, b \in \{\alpha, \beta\}\}$.
- The rendezvous letters $\mathcal{A}^f$ are $\mathcal{A}_r \cup (\mathcal{S} \times \{\alpha, \beta\}) \cup \left(\overline{\mathcal{S} \times \{\alpha, \beta\}}\right)$, where $\overline{\mathcal{S} \times \{\alpha, \beta\}} = \left\{\overline{s, b} : s \in \mathcal{S}, b \in \{\alpha, \beta\}\right\}$.
  The rendezvous labels are $\mathcal{A}_r \times [2K] \cup \left(\mathcal{S} \times \{\alpha, \beta\} \cup \overline{\mathcal{S} \times \{\alpha, \beta\}}\right) \times [2]$.
- The transitions $\mathcal{T}^f$ are as follows:

Original $t \in \tilde{\mathcal{T}}$	New transitions in $\mathcal{T}^f$	for
$t : s \xrightarrow{b} s'$	$(s, A; b') \xrightarrow{s,b;2} (s', b; b')$	$b, b' \in \{\alpha, \beta\}$
Broadcast	$(s, A; b) \xrightarrow{s,b;1} (s, b; b)$	$b \neq b'$
	$(s, b; b) \xrightarrow{\beta} (s, B; b)$	
	$(s, B; b) \xrightarrow{\overline{s,b};1} (s', A; b)$	
	$(s, b; b') \xrightarrow{\overline{s,b};2} (s', A; b')$	
$t : s \xrightarrow{a,k} s'$	$(s, A; \alpha) \xrightarrow{a,2k-1} (s', d; \alpha)$	
Rendezvous	$(s, A; \beta) \xrightarrow{a,2k} (s', d; \beta)$	
$s \in \tilde{\mathcal{S}}$	$(s, d; b) \xrightarrow{b} (Z, b)$	$d \neq b, b \in \{\alpha, \beta\}$
New Sink	$(Z, b) \xrightarrow{b} (Z, b)$	$d \in \{\alpha, \beta, A, B\}$

Each original process of P is simulated by a pair of processes in P^f, one in group α, one in group β. They can take rendezvous transitions together, as long as they are in a state with A. Before every broadcast transition, they have to guess it together, moving to a state with α or β. Then, one of them will be active in the group broadcast transition.

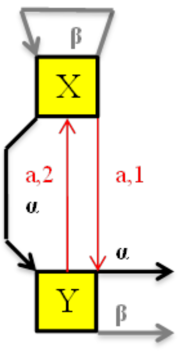

Figure 4.1: Part of a rendezvous-multiple broadcast network template

That process can tell this to its partner, moving both to a state with A again.

A faithful simulation of an original global run of P in P^f has the same rendezvous transition with twice as many processes participating. Before every broadcast, all processes guess it, and after it all processes move back to a state with A.

Figure 4.2 shows the RFGBN constructed from the rendezvous-multiple broadcast network from Figure 4.1. Figure 4.1 is an incomplete network, otherwise the resulting RFGBN would be too complicated to show in a graphic.
In Figure 4.2 the left side shows group α and the right side group β. The states simulating the state X from Figure 4.1 are in the upper half, the ones simulating Y in the lower half. For simplicity, the sink Z is missing from Figure 4.2. All states that do not have a group broadcast transition starting in it have their respective group broadcast transition leading to the sink Z.

When any process is active in a group broadcast it did not guess correctly, it sinks.

In order to prove the correctness of the simulation we will look at its counter representation. The following lemma states that as long as no process sinks, there are always as many processes in an state of group α as there are in the corresponding state of group β.

Lemma 6. *Let $\vec{\pi}$ be a global run of $\mathcal{P}^f_M$ with an infinite number of α and β group broadcasts. Let $S_i(s,d;d')$ be the number of processes in state (s,d) and group d' after the i-th transition. Then, there exists an index I, such that for all configurations $i \geq I$:*

- $S_i(s,\alpha;\alpha) + S_i(s,B;\alpha) = S_i(s,\alpha;\beta)$
- $S_i(s,\beta;\beta) + S_i(s,B;\beta) = S_i(s,\beta;\alpha)$

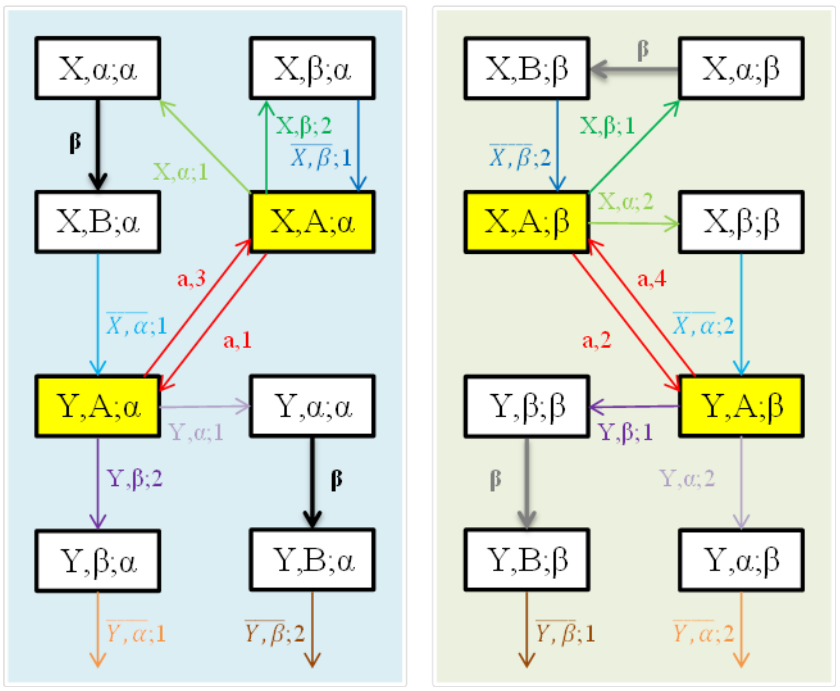

Figure 4.2: RFGBN template to simulate the multiple broadcast network in Figure 4.1

- $S_i(s, A; \alpha) = S_i(s, A; \beta)$

As there are only a finite number of processes which can sink and $\vec{\pi}$ is an infinite run there must exist an index I, after which no process sinks. Let us first establish that $S_i(s, \alpha; \alpha) + S_i(s, B; \alpha) - S_i(s, \alpha; \beta)$ stays constant for $i \geq I$. Processes can enter and leave these states only pairwise. The only way a process from one side can leave while the other one stays is by sinking. If the difference is not 0, there are always processes on one side of the equation. Then it takes at most two group broadcasts in that group for processes to sink. As there are an infinite number of both group broadcasts, that will happen. Therefore $S_i(s, \alpha; \alpha) + S_i(s, B; \alpha) = S_i(s, \alpha; \beta)$ for all $i \geq I$. The other two properties can be shown the same way. $\square$

The consequence of lemma 6 is that starting in I, every global run $\vec{\pi}$ of the RFGBN $\mathcal{P}^f$ simulates a global run of $\mathcal{P}$ faithfully. The simulated path is not necessarily initial. The following theorem concludes this chapter:

Theorem 4. *The following PMCP is undecidable: Given a RFBGN template* $P = \langle \mathcal{S} \times [L], \mathcal{S}_0, \mathcal{A} \times [K] \cup \{\beta\}, \mathcal{T} \rangle$, *a set of initial states* $\mathcal{S}_0' \subseteq \mathcal{S}_0$ *and a set undesired states* $\mathcal{Z} \subset \mathcal{S}$, *do all infinite executions starting in* $\mathcal{S}_0$ *reach a state in* $\mathcal{Z}$*?*

This follows from the reduction to RMBN. Let $\tilde{P}^f$ be the RFGBN from construction 6. The set of undesired states is $\{(Z,\alpha),(Z,\beta)\} \cup \Big\{(s,d;b) \in \mathcal{S}^f \times \{\alpha,\beta\} : s \in \mathcal{Z}\Big\}$ and the set of starting states only contains $(({}_0s,0),A;\alpha)$. If the question in Theorem 4 would be decidable, so would be the question in Theorem 3 due to Lemma 6.

CHAPTER 5

Unknown number of groups

In all models presented so far, the PMCP for was undecidable for liveliness specifications. This section introduces another special case of an RMBN. As in Chapter 4, the process are arranged in groups. Broadcasts only affect processes in one group, but rendezvous can connect different groups.
The difference however is that the number of groups in not fixed by the template. At the beginning of the run, the processes are assigned to any number of groups. Rendezvous transitions do not mention specific groups. Instead, they only set the relative group positioning of the processes taking different roles.

5.1 Formal definition

Definition 9. P: Template for Rendezvous-Group Broadcast Networks

The process template for RGBN is the RMBN template $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G} \cup \{\beta\}, \mathcal{T} \rangle$, where $\mathcal{A}$ is set of rendezvous letters, $K \in \mathbb{N}$ and $\mathcal{G}$ is a set of group partition symbols. All transitions t with $label(t) = \beta$ are broadcast transitions, the others are rendezvous transitions. Rendezvous transitions are written in the form $s \xrightarrow[g]{a,k} s'$, with $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $k \leq K$ and $g \in \mathcal{G}$.
$ltr(s \xrightarrow[g]{a,k} s')$ refers to a, $role(s \xrightarrow[g]{a,k} s')$ to k and $grp(s \xrightarrow[g]{a,k} s')$ to g.
Given a rendezvous letter a, $grp_a(k)$ is the group partition symbol associated to role k for the letter a. It is the group partition symbol g used in the unique transition with letter a and role k.

The group partition symbols $g \in \mathcal{G}$ should be understood as an partitioning of the roles for each letter. The processes taking the roles in the set $\{k \leq K : grp_a(k) = g\}$ need to be in the same group in a rendezvous transition with letter a.

Definition 10. The network $P^N = \left\langle \mathcal{S}^N \times [N]^N, \mathcal{S}_0^N \times [N]^N, \mathcal{A} \times [N]^K \cup \{\beta\} \times [N], \mathcal{T}^N \right\rangle$.
W.l.o.g. it can be assumed that every configuration $(\vec{s}, \vec{l})$ is in $\mathcal{S}^N \times [L]^N$, where $L \leq N$ is the number of groups and there is at least one process in every group $l \leq L$.

As for RFGBN in Construction 5, the global group broadcast transitions are elements of $\left(\mathcal{S}^N \times [L]^N\right) \times (\{\beta\} \times [L]) \times \left(\mathcal{S}^N \times [L]^N\right)$.
Written in the form $(\vec{s}, \vec{l}) \xrightarrow{\beta,l} (\vec{s'}, \vec{l})$, only processes in group l are active:
$\vec{s}(n) = \vec{s'}(n) \qquad \vec{l}(n) \neq l$
$\vec{s}(n) \xrightarrow{\beta} \vec{s'}(n) \in \mathcal{T} \qquad \vec{l}(n) = l$
Note that the broadcast transitions are not group specific in the templates of this model.

As in the general RMBN, the global rendezvous transition are elements of $\left(\mathcal{S}^N \times [N]^N\right) \times \left(\mathcal{A} \times [N]^K\right) \times \left(\mathcal{S}^N \times [N]^N\right)$. There is one additional condition for global rendezvous transitions written in the form $(\vec{s}, \vec{l}) \xrightarrow{a,\vec{p}} (\vec{s}, \vec{l})$, ensuring that roles in the same set of the portioning are taken by processes in the same group:

$$\begin{array}{ll} \vec{s}(n) = \vec{s'}(n) & \forall n \notin range(\vec{p}) \\ \vec{s}(\vec{p}(k)) \xrightarrow[grp_a(k)]{a,k} \vec{s'}(\vec{p}(k)) \in \mathcal{T} & \forall k \leq K \\ \vec{l}(\vec{p}(k)) = \vec{l}(\vec{p}(k')) & \forall k, k' \leq K : grp_a(k) = grp_a(k') \end{array}$$

For global rendezvous transitions $\vec{t} = (\vec{s}, \vec{l}) \xrightarrow{a,\vec{p}} (\vec{s'}, \vec{l})$, $label(\vec{t}) = (a, \vec{p})$, $src(\vec{t}) = (\vec{s}, \vec{l})$ and $dst(\vec{t}) = (\vec{s}, \vec{l})$, as inherited from being an LTS. To access the individual elements, $ltr(\vec{t}) = a$ and $role(\vec{t}) = \vec{p}$, $src\_st(\vec{t}) = \vec{s}$, $dst\_st(\vec{t}) = \vec{s'}$ and $grp(\vec{t}) = \vec{l}$ can be used.
For a global group broadcast transition $\vec{t} = (\vec{s}, \vec{l}) \xrightarrow{\beta,l} (\vec{s'}, \vec{l})$, $ltr(\vec{t}) = \beta$, $role(t) = l$. Other access functions are defined as for global rendezvous transitions.
For a global transition $\vec{t}$ $\vec{t}(n)$ gives the transition process n takes. For a global rendezvous transition, that is $\perp$, if $n \notin range(f)$ and $\vec{s}(n) \xrightarrow[grp_a(k)]{a,k} \vec{s'}(n)$ if $n = \vec{p}(k)$.
For a global group broadcast transition $\vec{t}(n) = \perp$ if $\vec{p}(n) \neq i$. Otherwise, $\vec{t}(n) = \vec{s}(n) \xrightarrow{\beta} \vec{s'}(n)$. For both kinds of transitions, process n is called *active* in transition $\vec{t}$, if $\vec{t}(n) \neq \perp$.

Note that transitions with different group partition symbols can be in the same group. While this definition is less natural, it does simplify the reasoning about these networks. The expressive power is not changed by allowing this: Every run that is an execution if transitions from different partitions can be in the same group, is also an execution if they have to be in different groups. This is due to the fact that runs can be copied (see Lemma 7) to have several times the number of groups, all copies taking the same transitions. Then, the partitions of the roles of the global rendezvous transition can be distributed across the copies.

5.2 Reachability

In this section an algorithm will be provided, which solves the PCME problem for regular specifications. In order to do so, we construct the reachability-unwinding template $P^{-\circ}$. First, two lemmas will be presented, which allow the construction of runs with many processes from runs with fewer processes.

Lemma 7. *Let ${}_m\vec{\pi}, m \in [M]$ be a family of M global runs. Let $\mathcal{L}_m$ be the groups of ${}_m\vec{\pi}$ respectively. Let L be the number of groups of the merged run. Let $h_m : \mathcal{L}_m \to [L]$ be functions embedding the groups of the individual runs into $[L]$. Let ${}_m\tau$ be the sequence of groups in the order they have group broadcast transitions in ${}_m\vec{\pi}$.*
We call the broadcast comparable if the following is true: There is a sequence of groups in $[L]$, τ, representing the group broadcast of the merged run. τ has the property that for all m, $h_m({}_m\tau)$ is the sub-sequence of τ containing all groups in $range(h_m)$.
If the broadcasts are comparable, then there exists a merged run $\vec{\pi}$ with:

$$grp(\vec{\pi})\left(\sum_{\tilde{m}=1}^{m-1} dim({}_{\tilde{m}}\vec{\pi}) + n\right) = h_m(group({}_m\vec{\pi})(n)) \tag{5.2.1}$$

and

$$proj(\vec{\pi}, \sum_{\tilde{m}=1}^{m-1} dim({}_{\tilde{m}}\vec{\pi}) + n) = proj({}_m\vec{\pi}, n)$$

The proof of Lemma 7 is the same as for Lemma 2. Here, process from one run ${}_m\vec{\pi}$ are not concerned by broadcast in groups out of range of h_m.

Creating the reachability-unwinding template

In a similar manner as in [ARZS14, p. 5-6], and Algorithm 1, the reachability-unwinding template $P^{-\circ}$ can be calculated. As there is only one broadcast letter, the components are arranged in a lasso and can be identified by their index in the lasso. Inter-group rendezvous transitions yield an additional challenge compared to [ARZS14]. When a process takes a rendezvous transition with processes from other groups, those groups might have seen more or less group broadcast transitions. To overcome this challenge, $P^{-\circ}$ can be calculated iteratively. All components are calculated in each iteration. $comp(j, q)$ stands for the j-th component in the q-th iteration of the calculation. The first iteration only contains the states reachable without using any inter-group rendezvous transition. Every other iteration allows inter-group rendezvous transitions, as long as the source state of ever transition with a group symbol different from the observed transition is present in the previous iteration.

Note, that Chapter 6, which gives an generalization of this model where processes are capable of group changes uses the same proof with some adaptations for the added functionality of the model.

Algorithm 2. Construction of the unwinding template $P^{\multimap}$ from the RGBN template P

Given a RGBN template $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G} \cup \{\beta\}, \mathcal{T} \rangle$ the component $comp(j,q)$ is the LTS $\left\langle {}^{j}_{q}\mathcal{S}, {}^{j}_{q}\mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G}, {}^{j}_{q}\mathcal{T} \right\rangle$. ${}^{j}_{q}\mathcal{S}$ are subsets of $\mathcal{S} \times \{j\}$.

For the first iteration, the calculation for $comp(0,0)$ starts as the LTS
$\left\langle \mathcal{S}_0 \times \{j\}, \mathcal{S}_0 \times \{j\}, \mathcal{A} \times \mathbb{N}^{\mathcal{G}} \times [K] \times \mathcal{G}, \emptyset \right\rangle$
and the calculation of $comp(j,0)$ for $j \geq 1$ starts as
$\left\langle reach(comp(j-1,0),\beta) \times \{j\}, reach(comp(j-1,0),\beta) \times \{j\}, \mathcal{A} \times \mathbb{N}^{\mathcal{G}} \times [K] \times \mathcal{G}, \emptyset \right\rangle$, where *reach* is defined as in Algorithm 1.
For all other iterations, $comp(0,q)$ for $q \geq 1$ starts as
$\left\langle {}^{0}_{q-1}\mathcal{S}, \mathcal{S}_0 \times \{j\}, \mathcal{A} \times \mathbb{N}^{\mathcal{G}} \times [K] \times \mathcal{G}, {}^{j}_{q-1}\mathcal{T} \right\rangle$
and $comp(j,q)$ with, $j,q \geq 1$ start as
$\left\langle {}^{j}_{q-1}\mathcal{S} \cup (reach(comp(j-1,q),\beta) \times \{j\}), reach(comp(j-1,q),\beta) \times \{j\}, \mathcal{A} \times \mathbb{N}^{\mathcal{G}} \times [K] \times \mathcal{G}, {}^{j}_{q-1}\mathcal{T} \right\rangle$.

For $q = 0$, $comp(j,0)$ is calculated as in Algorithm 1, only using those rendezvous letters $a \in \mathcal{A}$, where all transitions labeled with a use the same group partition symbol. As in Algorithm 1, transitions and destination states are added to the component, if all required source states are present in the component.
When $reach(comp(J_{q-1},q)) = {}^{j}_{q}\mathcal{S}_0$ for some $j \leq J$, then the calculation of this iteration is finished. In this case, J_q is called the *number of components* of iteration q and $\tilde{J}_q = J_q - j$ is called the *cycle length* of the iteration. This is due to $comp(j,q) = comp(j - m\tilde{J}_q, q)$ for $J_q \geq j < J_q + m\tilde{J}_q$.

The number of components and the cycle length can vary from iteration to iteration. When $j > J_{q-1}$, then ${}^{j-(m \cdot \tilde{J}_{q-1})}_{q-1}\mathcal{S}$ is used instead of ${}^{j-m\tilde{J}_{q-1}}_{q-1}\mathcal{S}$.

For $q \geq 1$, the calculation of $comp(j,q)$ also includes inter-group rendezvous transitions. Let ${}_{g}\mathcal{S}_a$ / ${}_{g}\mathcal{S}'_a$ be the sets of states s for which there is a transition with source/destination s, letter a and group partition symbol g. While Algorithm 1 iteratively goes through rendezvous letters and checks, whether they should be added, we now go through triples $(a, g, \vec{g})$, where $a \in \mathcal{A}$, $g \in \mathcal{G}$ for which there is a transition with letter a and group partition symbol g and $\vec{g} \in ([J_{q-1}] \cup j)^{\mathcal{G}}$, with $\vec{g}(g) = j$ and $\vec{g}(\hat{g}) \in [J_{q-1}]$ for $\hat{g} \neq G$. In order for the triple $(a, g, \vec{g})$ to be added to $comp(j,q)$, copies of all states in ${}_{g}\mathcal{S}_a$ must be present in $comp(j,q)$ and for all other group symbols $\hat{g}$, the copies of the states ${}_{\hat{g}}\mathcal{S}_a$ must be present in the component of the previous calculation $comp(\vec{g}(\hat{g}), q-1)$. Then, copies of all states in ${}_{g}\mathcal{S}'_a$ are added to ${}^{j}_{q}\mathcal{S}$, and the rendezvous transition of the form $(s,j) \xrightarrow[g]{a,\vec{g},k} (s',j)$ are added to ${}^{j}_{q}\mathcal{T}$ for all transitions with letter a and group symbol g.

In each iteration, states and transitions are added to components, never removed. A fixed point is reached, when no new states were added in the last iteration Q. The components $comp(j,Q)$ for all $j \leq J_Q$ are then combined to form the unwinding template $P^{\multimap}$. Broadcast transitions are added as in Algorithm 1.

$J = J_q$ is the number of components of the unwinding template $P^{\multimap}$ and $\tilde{J} = \tilde{J}_Q$ is its cycle length.

Properties of the reachability-unwinding template

This section will highlight a couple of important properties of the template $P^{\multimap}$.

The following lemma allows loading as many processes as needed into one group:

Lemma 8. *Let $P^{\multimap}$ be a reachability-unwinding template of P with J components and a cycle length of $\tilde{J}$. For $j \in \mathbb{N}$, let $\hat{j}$ be $j - m\tilde{J}$ for the minimal $m \geq 0$, such that $\hat{j} \leq J$. Let ${}^{\hat{j}}s$ be a state in component $comp(\hat{j})$.*

Then, there exists a global run $\vec{\pi}$ of $P^{\multimap}$ with j group broadcasts in the observed group and Process 1 ends the run in state s.

Proof:
The proof is based on the proof of Lemma 3, with the following differences:

The outer hypothesis is over j instead of τ.

There is a third layer of induction, over the iterations of Algorithm 2. For the case of a rendezvous transition added in iteration q, it is assumed that Lemma 8 is true for all states added in iterations $< q$ for any component $j' \leq J$. When merging the runs, the observed groups of the runs for states that are source states for transitions with the same group symbol are merged. They are comparable, as these have the same number of group broadcast transitions in the observed group, which must be their component number.

Lemma 8, in combination with Lemma 7, allows loading any number of processes into states of the corresponding component for any number of group broadcast in any number of groups.

While ${}^{\circledcirc}$ can be defined for RGBN as for RMBN, attributing components to transitions and runs is more complicated. For global runs, the component of each group is determined by the number of broadcasts in that group, allowing to define $\vec{\pi}^{\multimap}$ for a global run $\vec{\pi}$. For local runs, only the component of the observed group is determined. The rendezvous letters of runs in the unwinding template guess the component of the other groups involved in rendezvous transitions. Therefore, ${}^{\multimap}$ can not be defined for local runs. Therefore, Theorem 5 is only formulated in terms of ${}^{\circledcirc}$.

Theorem 5. *Let P be a rendezvous-group broadcast network template and $P^{\multimap}$ its unwinding template. If π is an finite run of P, then π is an execution if there is an finite run $\hat{\pi}$ of $P^{\multimap}$ with $\hat{\pi}^{\circledcirc} = \pi$.*
Every finite run of $P^{\multimap}$ is an execution.

The same proof as for Theorem 1 is used. The main difference is that $\hat{\pi}$ is used instead of $\vec{\pi}^{-\circ}$ It has to be constructed for $\Rightarrow$. There must be a global run $\vec{\pi}$ with $proj(\vec{\pi}, 1) = \pi$. Then, $\hat{\pi} = proj(\vec{\pi}^{-\circ}, 1)$

5.3 Infinite runs

The goal of this section is proving that the PMCP is decidable for infinite runs of rendezvous-group broadcast networks and ωB-regular specifications. This is achieved by constructing the B-automaton B_P^∞, which accepts the executions of the network defined by the template P. The main building block of the automaton B_P^∞ are transition colors, which are defined as follows:

Definition 11. Color

For an RGBN template P the color of a transition in the unwinding template $P^{-\circ}$ is:

- red, if it can only occur finitely many times in each group.
- blue, if it can occur infinitely many times in groups with finitely many group broadcasts, but only a finite number of times in groups with an infinite number of group broadcasts
- orange, if it can occur only a finite number of times in groups with finitely many group broadcasts and it can occur infinitely many times in groups with an infinite number of group broadcasts, but only a bounded number of times between each group broadcast.
- green, if it can occur infinitely many times in any group and an unbounded number of times between group broadcasts.
- violet otherwise

These colors are basically the same as the colors of transitions in [ARZS14]. The main difference is that both orange and violet transitions are called orange in that work. They distinguish the two cases by calling what we call violet to be orange and 'locally reusable'.

We will prove that the following B-automaton describes the infinite executions of RGBN. It recognizes the same words as the B-automaton for RBN in [ARZS14]. The basic idea is that it consists of three copies of $P^{-\circ}$. One for the finite initial part of infinite words. The set of transitions within this copy, ${}^\infty\mathcal{T}^{ini}$ contains all transitions from $P^{-\circ}$.
The second copy is used for the case of Process 1 going through a finite number of group broadcasts. The transitions of this copy ${}^\infty\mathcal{T}^{noB}$ only contains blue, green and violet transitions.

The second copy is used for the case of Process 1 going through an infinite number of group broadcasts. The transitions of this copy ${}^{\infty}\mathcal{T}^{infB}$ only contains orange, green and violet transitions.
The set of transitions ${}^{\infty}\mathcal{T}^{con}$ connects the first copy with the other two.

Construction 7. B-automaton B_P^{∞} accepting the executions of P.

Let $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G} \cup \{\beta\}, \mathcal{T} \rangle$ be a rendezvous-group broadcast network and $P^{\multimap} = \langle \mathcal{S}^{\multimap}, \mathcal{S}_0^{\multimap}, \mathcal{A} \times [K] \times \mathcal{G} \cup \{\beta\}, \mathcal{T}^{\multimap} \rangle$ its unwinding template. Then, $B_P^{\infty} = \langle {}^{\infty}\mathcal{S}, {}^{\infty}\mathcal{S}_0, \mathcal{T} \times (\{\gamma, 0\} \times \mathbb{N}), {}^{\infty}\mathcal{T}, \Phi \rangle$ is defined as:

- The set of states ${}^{\infty}\mathcal{S}^{\infty}$ is $\{(d, s, j) | d \in \{ini, infB, noB\}, {}^{j}s \in \mathcal{S}^{\multimap}\}$
- The set of initial states ${}^{\infty}\mathcal{S}_0 = \{(ini, s, 0) | {}^{0}s \in {}^{o}\mathcal{S}_0\}$
- The input alphabet are the original transitions $\mathcal{T}$ of the template P.
- There is one counter γ.
- The transitions ${}^{\infty}\mathcal{T}$ consist of four parts ${}^{\infty}\mathcal{T}^{ini} \cup {}^{\infty}\mathcal{T}^{infB} \cup {}^{\infty}\mathcal{T}^{noB} \cup {}^{\infty}\mathcal{T}^{con}$, one for each type of the states and one connecting the types.
  For all rendezvous transitions $t : ({}^{j}s \xrightarrow[G]{a,k} {}^{j}s') \in {}^{o}\mathcal{T}$, these four parts contain the following transitions:

Part	Transition	Condition
${}^{\infty}\mathcal{T}^{ini}$	$(ini, s, j) \xrightarrow[\gamma:=\gamma+1]{t} (ini, s', j)$	All $t \in {}^{o}\mathcal{T}$
${}^{\infty}\mathcal{T}^{infB}$	$(infB, s, j) \xrightarrow[\gamma:=\gamma+1]{t} (infB, s', j)$	t is orange or violet
	$(infB, s, j) \xrightarrow{t} (infB, s', j)$	t is green
${}^{\infty}\mathcal{T}^{noB}$	$(noB, s, j) \xrightarrow{t} (noB, s', j)$	t is blue, green or violet
${}^{\infty}\mathcal{T}^{con}$	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j)$	All $t \in {}^{o}\mathcal{T}$
	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (noB, s', j)$	All $t \in {}^{o}\mathcal{T}$

  For all group broadcast transitions $t : ({}^{j}s \xrightarrow{\beta} {}^{j'}s') \in {}^{o}\mathcal{T}$, there are no transitions in ${}^{\infty}\mathcal{T}^{noB}$. The other parts contain the following transitions:

Part	Transition	Condition
${}^{\infty}\mathcal{T}^{ini}$	$(ini, s, j) \xrightarrow[\gamma:=\gamma+1]{t} (ini, s', j')$	All $t \in {}^{o}\mathcal{T}$
${}^{\infty}\mathcal{T}^{infB}$	$(infB, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j')$	t is orange
${}^{\infty}\mathcal{T}^{con}$	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j')$	All $t \in {}^{o}\mathcal{T}$
	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (noB, s', j')$	All $t \in {}^{o}\mathcal{T}$

- The acceptance condition Φ states that $\limsup \gamma < \infty$.

Theorem 6. *For a RGBN template P the language accepted by B_P^∞ from Construction 7 is $exec^{inf}(P)$.*

$\supseteq$:
Let π be an execution of P and $\pi^{-\circ}$ the corresponding execution in the unwinding template $P^{-\circ}$. Due to the way colors are defined in Definition 11, the following must be true: If $\pi^{-\circ}$ contains a finite number of group broadcast transitions, there must exist an index $i \in \mathbb{N}$ after which all transitions are blue, violet or green. If it contains an infinite number of group broadcast transitions, there must exist an index $i \in \mathbb{N}$ after which all transitions are orange, violet or green. Observe that copies of all transitions in $P^{-\circ}$ are present in ${}^\infty\mathcal{T}^{ini}$ and ${}^\infty\mathcal{T}^{con}$. ${}^\infty\mathcal{T}^{noB}$ contains copies all blue, violet and green transitions while ${}^\infty\mathcal{T}^{infB}$ contains copies of all orange, green or violet transitions.

We can therefore construct a run π^B of B_P^∞ as follows: It contains the copies of the transition $\pi^{-\circ}$ in ${}^\infty\mathcal{T}^{ini}$ until index i. Then, the copy of transition $\pi_i^{-\circ}$ in ${}^\infty\mathcal{T}^{con}$ connecting to the state (d, s, j) with $d = noB$ is used, if $\pi^{-\circ}$ contains a finite number of group broadcast transitions. Otherwise, the transition in ${}^\infty\mathcal{T}^{con}$ connecting to the state with $d = infB$ is used. From there on, π^B contains copies of the transition of $\pi^{-\circ}$ in ${}^\infty\mathcal{T}^{noB}$, if it contains a finite number of group broadcast transitions and copies in ${}^\infty\mathcal{T}^{infB}$ otherwise.

It only remains to show that the counter γ does not prevent the acceptance of π^B. π^B stays in ${}^\infty\mathcal{T}^{ini}$ only up to transition I. Therefore, counter γ is bounded by I during these transitions. If π^B then moves on to transitions in ${}^\infty\mathcal{T}^{noB}$, the counter will never be increase again.
Otherwise, it will be increased with every orange or violet rendezvous transition and reset with every group broadcast transition. Let $\mathcal{T}_{ov}$ be the set of all orange and violet rendezvous transitions in $P^{-\circ}$. By Definition 11, for every transition $t \in \mathcal{T}_{ov}$ the number of occurrences of t between two group broadcast transitions must be bounded by a natural number c_t. The counter γ is therefore bounded by $\sum_{t \in \mathcal{T}_{ov}} c_t$.

The remainder of Chapter 5.3 will be used to prove the other direction $\subseteq$. More machinery is required to do so. First, we will define configuration isomorphisms and pseudo-cycles:

Definition 12. Configuration Isomorphism f

Let $(\vec{s}, \vec{l})$ and $(\vec{s}', \vec{l}')$ be two configurations with n. Then $f : [N] \to [N]$ is a configuration isomorphism between $(\vec{s}, \vec{l})$ and $(\vec{s}', \vec{l}')$ iff it is bijective and $\vec{s}(n) = \vec{s}'(f(n))$ for all $n \leq N$ and $\vec{l}(n) = \vec{l}(n')$ iff $\vec{l}'(f(n)) = \vec{l}'(f(n'))$ for all $n, n' \leq N$.

A isomorphism exists between two configurations, if they contain the same number of groups and for each group in $(\vec{s}, \vec{l})$, there is a group in $(\vec{s}', \vec{l}')$, containing the same number of processes in each group.
Observe that two counter represented configurations are isomorphic, if $ctr(\vec{s}, \vec{l})$ contains the same columns as $ctr(\vec{s}', \vec{l}')$, but potentially in a different ordering.

Definition 13. Pseudo-Cycle

A pseudo-cycle of a group broadcast template P with N processes is a not necessarily initial finite global path C of P^{∞} for which there is a configuration isomorphism between $src(C)$ and $dst(C)$.

For a given isomorphism f between $src(C)$ and $dst(C)$, a group l is said to *transform* into group l' in a pseudo-cycle if for some $n \leq N$ $grp(C)(n) = l$ and $grp(C)(f(n)) = l'$.

Intuitively, a pseudo-cycle starts and ends in basically the same configuration, but individual processes and individual groups are in the situation of different processes and groups in the end.
Observe that 'transforms into' is a permutation of the groups of π, which allows the following definition:

Definition 14. Group Transformation Cycles

The cycles of the permutation 'transforms into' are called *Group Transformation Cycles*.

Note that transforming into and group transformation cycles depend on the specific isomorphism f. Group Transformation Cycles are used to characterize Groups:

Definition 15. The *type* of a group l in a pseudo-cycle is:

- *with broadcasts*, if at least one global broadcast transition is in group l;
- *between broadcasts*, if no broadcast is in group l, but there is at least one group with broadcasts in the group transformation cycle of l;
- *without broadcasts* otherwise.

Note that as there is a fixed finite number of states and groups in each run, in any infinite run there must be a configuration appearing infinitely many times. When a transition appears an infinite number of times in an execution, it must appear on a pseudo-cycle. This is because the transition must appear between two instances of the same configuration on a global run the execution is the projection of. Such a pseudo-cycle is called a *witness* of the color of the transition. If the execution contains an infinite number of group broadcast transitions, then the witnessing pseudo-cycle contains a group broadcast transition in the same group as the transition the pseudo-cycle is witness of. This group is called the *observed* group. We can therefore conclude the following lemma:

Lemma 9. *A transition* $t : s \xrightarrow[g]{a,k} s'$ *of* $P^{\multimap}$ *is:*

1. *red, iff it does not appear on any pseudo-cycle of $P^{-\circ}$;*
2. *blue, iff it appears on a pseudo-cycle C_t^{noB} of $P^{-\circ}$ in a group without broadcasts, but it does not appear on any pseudo-cycle in a group with broadcasts;*
3. *orange, iff it appears on a pseudo-cycle C_t^{hasB} of $P^{-\circ}$ in a group with broadcasts, but it does not appear on any pseudo-cycle in a group with broadcasts.*
4. *green, iff it appears on a pseudo-cycle C_t^{noB} of $P^{-\circ}$ in the observed group l, which is with broadcasts, there is a pseudo-cycle C_t^{hasB} where the observed group l' is with group broadcast and the starting configurations of the observed group of C_t^{noB} is nested in the starting configuration of C_t^{hasB}. I.e. there is an injective function h_t from the processes of the observed group of C_t^{noB} to the processes of the observed group of C_t^{hasB} with $src(C_t^{noB})(n) = src(C_t^{hasB})(g_t(n))$.*
5. *violet, iff it appears on a counter represented pseudo-cycle C_t^{noB} of $P^{-\circ}$ in a group without broadcasts, and on a counter represented pseudo-cycle C_t^{hasB} in a group with broadcasts, but the condition for the starting configurations in (4) can not be met.*

For the pseudo-cycle C_t^d, d is called the *type* of the pseudo-cycle and t is the transitions the pseudo-cycle is witness of.

Equipped with the concept of witnessing pseudo-cycles we can now give the idea behind the proof of $\subseteq$ of Theorem 6. Given a run π accepted by B_P^∞, a global run $\vec{\pi}$ in $P^{-\circ}$ is constructed, such that the projection to Process 1 corresponds to π. This is done by running the witnessing pseudo-cycles for all transitions simultaneously and Process 1 swaps in and out of the position of the process that takes the transition the pseudo-cycles are witness for.
Let V be the upper limit of the counter in B_P^∞ for the run π. For orange and violet transitions there are V copies of the witnessing pseudo-cycle available. This way, the transition can happen up to V times between broadcasts and the pseudo-cycles get reset during the broadcast.

In order for Process 1 to swap between pseudo-cycles it is necessary for the observed groups of different pseudo-cycles to be merged into the same group. In order to do that, the pseudo-cycles must have their broadcast transitions at the same time (see Lemma 7). This is not the only condition we need to stitch the observed pseudo-cycles together to construct $\vec{\pi}$.

Lemma 10 lists the Conditions for all witnessing pseudo-cycles. We will then prove $\subseteq$ of Theorem 6 assuming that the pseudo-cycles from Lemma 9 fulfill all Conditions of Lemma 10. We will point out which condition is used when. This will give an understanding why these conditions are needed. Afterwards, we will prove Lemma 10. Note that the Conditions were not just created to make the stitching here work, but also to allow us to give an algorithm deciding the color of transitions. This algorithm will be described as last part of this section.

Lemma 10. *If there are witnessing pseudo-cycles $\hat{C}_t^d$ for the color of edges according to Lemma 9, then there are also witnessing pseudo-cycles C_t^d and a fixed configuration isomorphism f_t^d between the source and destination configuration of C_t^d, which additionally fulfill the following conditions:*

1. *Every group with broadcasts transforms into a group between broadcasts.*
2. *Every group between broadcasts transforms into a group with broadcasts.*
3. *Every group without broadcasts transforms into itself.*
4. *Every group with broadcasts has exactly one group broadcast transition in it.*
5. *All broadcast transitions happen after all rendezvous transitions.*
6. *For every kind of group $(j, e) \in [J] \times \{hasB, betB, noB\}$ there is exactly one group of kind (j, d). For green transitions t, C_t^{noB} has an additional second group of the same kind as the observed group.*
7. *For a rendezvous transition t, the first global transition of the pseudo-cycles witnessing t's color has Process 1 take t.*
8. *For the witnessing pseudo-cycles C_t^{hasB} for orange broadcast transitions, Process 1 is only active in one group broadcast transition and no rendezvous transition. Process $f_t^d(1)$ is not active in any transition.*

It will shortly be explained what the kind of a group is. Let it be noted that Conditions 4 and together imply Conditions 1 and 2. They are still stated as proving this fact is not necessary.

The following terms and operations are used for the construction of $\vec{\pi}$ to prove $\subseteq$ of Theorem 6:

- *Kind*: For a witnessing pseudo-cycle, the kind of a group refers to a tuple $(j, e) \in [J] \times \{hasB, betB, noB\}$, where J is the number of components in $P^{\to\circ}$. Both j and e can be derived from the pseudo-cycle given a group. j is the component the processes in the group are in in the source configuration of the pseudo-cycle. $e = hasB$, if the group is with broadcasts, $e = betB$ if it is between broadcasts and $e = noB$, if it is without broadcasts.
  j is called the component of the group and e its type. Condition 6 states that for every kind (j, e) there is exactly one group of kind (j, e), with the exception of the witnessing pseudo-cycle of green edges without broadcasts in the observed group, which has a second group of the same kind as the observed group.

- *Job*: In configurations of the global run $\vec{\pi}$, some groups have jobs. Each job is a triple $(j, e, d) \in [J] \times \{hasB, betB, noB\} \times \{hasB, noB\}$. In every configuration, there is exactly one group for each job and each group has at most one job. There may be groups without a job.
  j is called the component of the group, d its group type and e its cycle type.
- *Copy*: A copy of a pseudo-cycle is a tuple (C_t^d, v), where $v \in \mathbb{N}$. The number of copies depend on the transition:
  If t is an orange or violet rendezvous transition and $d = hasB$, then $v \leq V$.
  If t is a green rendezvous transition and $d = hasB$, then $v \leq 2$.
  Otherwise, there is only one copy $v = 1$.
- *Position*: In $\vec{\pi}$, processes have positions in configurations. These are triples $(C_t^d, \hat{n}, v)$, where (C_t^d, v) is a copy and $\hat{n}$ is process in the pseudo-cycle C_t^d.
  In every configuration, there is exactly one process in every position. Each process is in at most one position, there may be processes without a position.
- *Valid Position*: A position $(C_t^d, \hat{n}, v)$ of a process n in group with job (j, e, d') is called valid if The group of process $\hat{n}$ in C_t^d is of kind (j, e) and $d = d'$
- *Mark*: A copy of a pseudo-cycle (C_t^d, v) can be marked. That means that the first global transition, which involves the transition the pseudo-cycle is witness for (Condition 7), has already happened.
- *Valid Configuration*: A group with job is said to be in a valid configuration if the following is true for every process n in the group in a position $(C_t^d, \hat{n}, v)$:
  - The position is valid.
  - If (C_t^d, v) is unmarked, process n is in the same state as process $\hat{n}$ in the source configuration of C_t^d.
  - If (C_t^d, v) is marked, process n is in the same state as process $\hat{n}$ in reconfiguration after the first transition of C_t^d.
- *Basic Configuration*: A group is in a Basic Configuration, if it is in a valid configuration and no process in the group is in the position of a marked Copy, i.e. they are all in the source configuration of their pseudo-cycle.
- *Reset*: In a reset, all copies witnessing pseudo-cycles of type noB happen, except the first transition of marked pseudo-cycles. Every process with position in these cycles takes all the required rendezvous transition. In all pseudo-cycles, every rendezvous transition happens before all broadcast transitions (Condition 5). Every group with a job of the form $(j, hasB, noB)$ needs to now take exactly one group broadcast transition to finish all cycles of type noB (Conditions 4 and 5).
  At the end, there is a Position Change and a Job Change: Every process involved changes position according to f_t^d, a fixed configuration isomorphism between the source and destination configurations of C_t^d.

The job change works as follows: Only groups of the form $(j, betB, noB)$ and $(j, hasB, noB)$ change job. Groups with job $(j, betB, noB)$ now have the job $(j, hasB, noB)$ (Condition 2) and groups of the form $(j, hasB, noB)$ now have the form $(j', betB, noB)$ (Condition 1), where $j' = j+1$, if $j < J$ and $j' = J - \tilde{J} + 1$, if $j = J$ and $\tilde{J}$ is the cycle length of $P^{-\circ}$. As groups without group broadcasts transform into themselves (Condition 3), groups with jobs of type noB keep their job.
All copies of cycles of type noB get unmarked.

- *Flush*: In a flush, all copies of witnessing pseudo-cycles of type $hasB$ happen, except the first transition of marked pseudo-cycles. In order for every process involved to take the correct broadcast transition, there is a global broadcast transition in every group with job of the form $(j, hasB, hasB)$.
  Position change is again according to f^d_t.
  The job change works as follows: Only groups of the form $(j, betB, hasB)$ and $(j, hasB, hasB)$ change job. Groups with job $(j, betB, hasB)$ now have the job $(j, hasB, hasB)$ (Condition 2) and groups of the form $(j, hasB, hasB)$ now have the form $(j', betB, hasB)$ (Condition 1).
  All copies of cycles of type $hasB$ get unmarked.

Equipped with these terms, we can now define the global run $\vec{\pi}$.

For the finite part where π stays within ${}^{\infty}\mathcal{T}^{ini}$ and ${}^{\infty}\mathcal{T}^{con}$, Theorem 5 is used to get a global run, whose projection to Process 1 is the prefix of π. This global run is merged with a global run that for every job loads a basic configuration into a group using Lemmas 8 and 7. From there on, we do a case distinction on whether the rest of the run is in ${}^{\infty}\mathcal{T}^{noB}$ or ${}^{\infty}\mathcal{T}^{hasB}$.

If the run decides to transition into ${}^{\infty}\mathcal{T}^{noB}$, π remains in the same component for the rest of the run. In addition to the basic configuration, one additional process is loaded into every state of the component. That process does not have a position. For each remaining transition the constructed global run is extended as follows:
Process 1 is in the correct state to perform the next transition. Let t be the next transition. Process 1 swaps into the position $(C^{noB}_t, 1, 1)$, e.g. the position of the first process of the only copy of the pseudo-cycle witnessing t without broadcasts in the observed group. In C^{noB}_t, Process 1 takes transition t in the first global transition of C^{noB}_t (Condition 7). Therefore, the processes in position take the first transition of C^{noB}_t and the copy $(C^{noB}_t, 1)$ gets marked. Now, the additional Process without position in the state Process 1 is now in is used, as Process 1 swaps out of Position. Then, there is a reset. This yields a configuration, where all groups are in a basic configuration and the group Process 1 is an has an additional process in every state of the component Process 1 is in.

The case of π moving into transitions in ${}^{\infty}\mathcal{T}^{hasB}$ is more complex.
The construction now depends on the kind of transition. Let t be the next transition in

π. Observe that after every step all groups are in a valid configuration.

- t is an orange or violet rendezvous transition:
  In this case π swaps into the position $(C_t^{hasB}, 1, \hat{v})$, where $\hat{v}$ is the smallest v, such that (C_t^{hasB}, v) is not marked. Then, the first transition of the copy $(C_t^{hasB}, \hat{v})$ happens and $(C_t^{hasB}, \hat{v})$ gets marked.

- t is a green rendezvous transition:
  In this case π swaps into the position $(C_t^{noB}, 1, 1)$. Note, that this is a *noB* cycle, despite being in a group of cycle-type *hasB*. Therefore, other processes in the same group need to swap position into the same group too. All processes with a position $(C_t^{noB}, \hat{n}, 1)$ in the observed group of C_t^{noB} swap position with a process in position $(C_t^{hasB}, n, 1)$ in the same state. Remember, that the green pseudo-cycle of type *noB* has two groups of that kind, one of them is the observed group.
  This is possible, as for green transitions the number of processes in $src(C_t^{noB})$ in the observed group in each state must be less or equal than the number of processes in $src(C_t^{hasB})$ in the observed group and the same state.
  This allows the observed group of C_t^{noB}, which is a group without group broadcast, to happen in a group of group-type *hasB*. After the first transition, the copy $(C_t^{hasB}, 1)$ Process 1 swaps out, into the position to take the next transition. If the next transition t' is green as well, it swaps into the position $(C_{t'}^{hasB}, 1, 2)$ instead. Now, a reset happens, where the observed group of C_t^{noB} happens in a group of cycle-type *hasB*. After the reset, the processes in that group swap back into the group of cycle-type *noB* making their positions valid again.

- t is a group broadcast transition:
  In this case, Process 1 moves into position $(C_t^{hasB}, 1, 1)$. Then, two flushes happen. That position is only active in one group broadcast transition and no rendezvous transition (Condition 8). The first flush causes Process 1 to swap position according to f_t^d. In the next flush, Process 1 is not active in any transition (Condition 8). This is possible, as the group it is now in is between broadcasts.

  After these flushes, all groups are in a basic configuration.

This construction creates a global run, which projected to Process 1 yields π. This finishes the proof of Theorem 6 up to the proof of Lemma 10.

Proof of 10: The first step is to ensure Condition 7: t needs to be part of the first transition. This can be achieved for non-green edges by shifting the start of the original cycle C_t^d right before the transition t appears in. Renaming the processes ensures that Process 1 takes t.

For green and violet transitions, the process above has to be done with both pseudo-cycles. For green transitions, a slight adaptation has to be used to ensure that the the structured pseudo-cycle $\hat{C}_t^{hasB}$ has more processes in each state of the source configuration than $\hat{C}_t^{noB}$. When shifting the start of $\hat{C}_t^{hasB}$ to the position before t, the part of $\hat{C}_t^{noB}$ before t appears has to be appended to $\hat{C}_t^{hasB}$ at the end and the rest of the pseudo-cycle $\hat{C}_t^{noB}$ is added at the beginning.

The next preparation step is to ensure that every group with broadcasts transforms into a group with broadcasts. There are no groups between broadcast for this step. This is achieved by going through the cycle $\hat{C}_t^d$. Let $\hat{f}_t^d$ be an configuration isomorphism of the source and destination configuration of C_t^d. When the system went through $\hat{C}_t^d$ once, it goes through it again where each process n is active in the transitions in which process $\hat{f}_t^d(n)$ was active in the first walk-through. If the number of times going through $\hat{C}_t^d$ is a common multiple of the cycle lengths of the group transformation cycles, each group transforms into itself. There are therefore pseudo-cycles $\tilde{C}_t^d$ witnessing the colors of transitions, that go through the pseudo-cycles $\hat{C}_t^d$ m times, where every group transforms into itself. $\tilde{C}_t^d$ does therefore not contain any groups between broadcasts. Using the same method, we can require additionally that in witnessing pseudo-cycles C_t^{hasB} for group broadcast transitions t Process 1 is active in at least two group broadcast transitions. This will be necessary to ensure Condition 8 later on.

In the next step the pseudo-cycle $\dot{C}_t^d$ is created, which has $2R$ processes for each process in $\tilde{C}_t^d$, where R is the number of broadcasts where that process is active. For Processes in groups without broadcasts only 1 copy is needed.
Each of the copies for processes in groups with group broadcast is in a different group, but corresponding copies of different processes from the same group are in the same group. $\dot{C}_t^d$ has the same transitions as $\tilde{C}_t^d$, but different copies of the original process are active for each transitions. In $\dot{C}_t^d$, the first copy of a process is active in all transitions the original process was active in $\tilde{C}_t^d$ up to and including the first group broadcast transition, where the original process in active. The second copy is in the destination state of the first copy for the whole run, not active in any transition. This is possible, because it is now in a group between broadcast. The third copy has the same source configuration and is active in all transitions after the first group broadcast where the original process is active, up to and including the second group broadcast of the original process. Only the copies with odd numbers are active in any transition, with the exception of the last copy. It is active in all transitions after the last group broadcast. These transitions after the last broadcast are the reason groups between broadcast are needed in structured pseudo-cycles.

Note $\dot{C}_t^d$ is still a pseudo-cycle with the configuration isomorphism $\dot{f}_t^d$. Let (n, r) be the r-th copy of process n. Then $\dot{f}_t^d(n, r) = \begin{cases} (n, r+1) & , \ r < 2R \\ (f_t^d(n), 1) & , \ r = 2R. \end{cases}$

$\dot{C}_t^d$ fulfills all Conditions except 5 and 6 .
The groups of copies with odd r are with broadcasts, the groups of copies with even r are between broadcasts. Based on this, each group can be attributed a kind. All groups of the same kind will be merged to create C_t^d. To do so, the group broadcast in groups

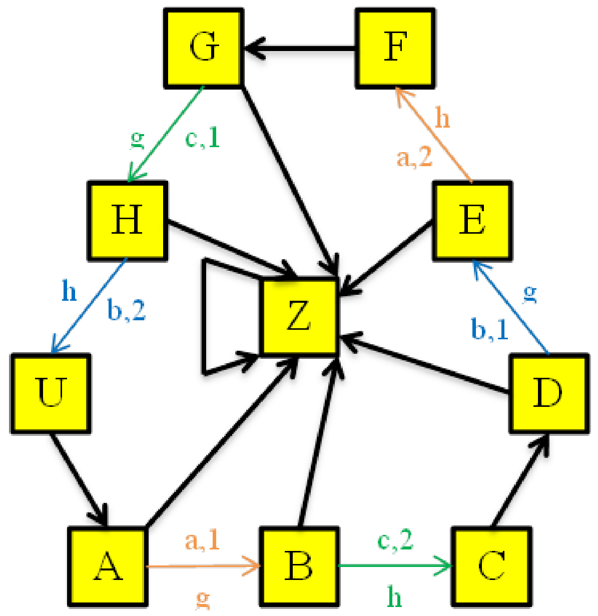

Figure 5.1: Rendezvous-group broadcast network template

to be merged must occur at the same time. In $\dot{C}_t^d$, all group broadcasts are the last transition where any process in that group is active. Therefore, all group broadcasts can be shifted to the end of the pseudo-cycle, after all rendezvous transitions. This allows all processes from groups of the same kind to be placed in the same group.
For green edges, the observed group of C_t^{noB} can not be merged with the other groups of the same kind. Otherwise, the condition of having fewer processes in each state than C_t^{hasB} can not be maintained.

C_t^d now fulfills all conditions 1 - 8. □

To illustrate this construction, the transformation of a witnessing cycle for an orange rendezvous transitions is illustrated in figures 5.1 to 5.3. Figure 5.1 shows a rendezvous-group broadcast network template without group change. All states are initial states. The unwinding template for this template has only one component and is isomorphic to the template itself. It contains three rendezvous letters a, b and c. They all describe pairwise inter-group rendezvous transitions.

Figure 5.2 shows a possible cycle $\hat{C}_t^{hasB}$ witnessing that transition $t = (A \xrightarrow[\Gamma]{a,1} B)$ appears on a pseudo-cycle with a group broadcast in the observed group, which implies it is orange, violet or green. (It actually is orange.)

Figure 5.3 shows the pseudo-cycle C_t^{hasB} constructed from $\hat{C}_t^{hasB}$ fulfilling Conditions 1 - 8. In $\hat{C}_t^{hasB}$, Process 1 is in Group 1, which has on group broadcast, and Process

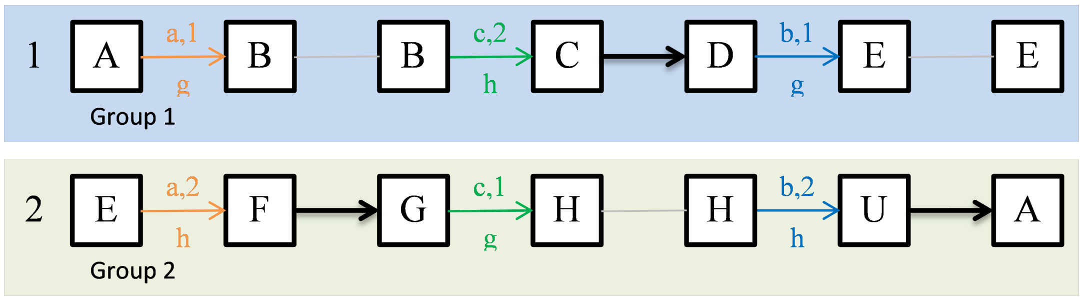

Figure 5.2: Possible cycle

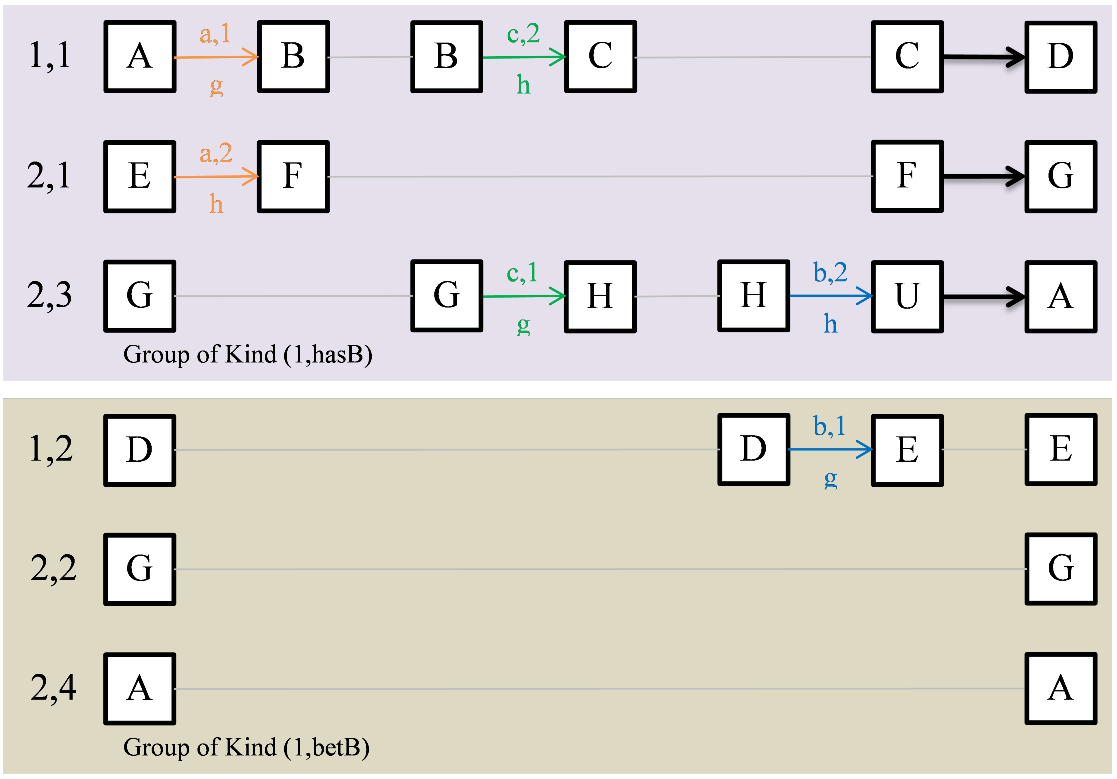

Figure 5.3: Pseudo-cycle according to Lemma 10 created from Figure 5.2.

2 in Group 2 has two group broadcasts. Therefore, the transitions of Process 1 are distributed among two processes in C_t^{hasB}, $(1,1)$ in the group of kind $(1, hasB)$ and $(1,2)$ an the group of kind $(1, betB)$. The transitions of Process 2 are distributed among four processes, two for both kinds $(1, hasB)$ and $(1, betB)$. As no group was without group broadcast in the original pseudo-cycle $\hat{C}_t^{hasB}$, the group of kind $(1, noB)$ is empty.

5.4 Deciding Colors

Calculating the colors of transitions $P^{\multimap}$ can be reduced to deciding colors for the case of one global broadcasts letter as presented in [ARZS14]. To do so, we construct the following RMBN with one broadcast letter β:

Construction 8. $\tilde{P}$: RMBN with one broadcast letter β from the RGBN $P^{\multimap}$ to decide its colors.

Given an RGBN $P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G} \cup \{\beta\}, \mathcal{T} \rangle$
and it's unwinding template $P^{\multimap} = \left\langle \mathcal{S}^{\multimap}, \mathcal{S}_0^{\multimap}, \mathcal{A} \times [K] \times [J]^{\mathcal{G}} \cup \{\beta\}, \mathcal{T}^{\multimap} \right\rangle$,
the RMBN $\tilde{P} = \left\langle \tilde{\mathcal{S}}, \tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{T}} \right\rangle$, where:

- The set of states $\tilde{\mathcal{S}}$ contains triples (s, j, e), where ${}^j s \in \mathcal{S}^{\multimap}$ and $e \in \{hasB, betB, noB, obs\}$. $e = obs$ is only needed to test for the color green.
- All states are initial states.
- The rendezvous labels $\tilde{\mathcal{A}}$ are elements of $\mathcal{A} \times ([J] \times \{hasB, betB, noB, obs\})^{\mathcal{G}} \times [K]$. The rendezvous letters are tuples (a, f) with $a \in \mathcal{A}$ and $f \in \times ([J] \times \{hasB, betB, noB, obs\})^{\mathcal{G}}$. β is the only broadcast label.
- The transitions are $\tilde{\mathcal{T}}$. Rendezvous transitions are written in the form $(s, j, e) \xrightarrow{a,f,k} (s', j, e)$. It is a transition in $\tilde{P}$, if $f(grp_a(k)) = (j, e)$ and ${}^j s \xrightarrow{a,f^1,k} {}^j s'$ is a rendezvous transition in $P^{\multimap}$, where $f^1(G) = j$ if $f(G) = (j, e)$ for some e, i.e. f^1 is the projection of f onto the first component.
  Remember that $grp_a(k)$ is the group partition symbol used in the transition with letter a and role k.
  Depending on e, the broadcast transitions are as follows:
  $(s, j, hasB) \xrightarrow{\beta} (s', j', betB)$ iff ${}^j s \xrightarrow{\beta} {}^{j'} s'$ is a broadcast transition in $P^{\multimap}$.
  $(s, j, betB) \xrightarrow{\beta} (s, j, hasB)$
  $(s, j, noB) \xrightarrow{\beta} (s, j, noB)$
  $(s, j, obs) \xrightarrow{\beta} (s, j, obs)$

The basic idea of this construction is to include the group into the states. Due to Condition 6, only one group of each kind is needed, except the additional observed group in pseudo-cycles for green transitions. In structured pseudo-cycles, all groups of type *hasB* have their group broadcast at the same time. These group broadcast can be therefore simulated with one global broadcast, which does not change the state of processes in *betB* and *noB* groups. It switches processes between groups of type *hasB* and *betB*.

Observe that Construction 8 does not preserve colors, but it still can be used to determine the colors of transitions. It would be possible to create an RBN with the same executions as any given RGBN and get all the results of this section via reduction. Doing so would require basically the same arguments as presented here and we chose to present the results in a different manner.

The colors of transitions can be calculated using Construction 8 via the following lemma:

Lemma 11. *Let $P^{\multimap}$ be the unwinding template of the RGBN P and let $\tilde{P}$ be the RMBN from Construction 8. Then, a rendezvous transition $t = ({}^{j}s \xrightarrow{a,f,k} {}^{j}s')$ of $P^{\multimap}$ appears on a pseudo-cycle if a transition of the form $(s, j, e) \xrightarrow{a,\tilde{f},k} (s', j, e)$ with $\tilde{f}^{1} = f$ appears on a pseudo-cycle of $\tilde{P}$ with a group broadcast for some e.*

- *If $e = hasB$, then t appears on a pseudo-cycle with broadcasts in the observed group.*
- *If $e = noB$, then t appears on a pseudo-cycle without broadcasts in the observed group.*
- *t is green, iff there are pseudo-cycles for both $e = hasB$ and $e = obs$, C_t^{hasB} and C_t^{noB}, and for every ${}^{j}s \in {}^{j}\mathcal{S}$, the number of processes in state (s, j, obs) in the source configuration of C_t^{noB} is less or equal than the number processes in state $(s, j, hasB)$ in the source configuration of C_t^{noB}.*

A broadcast transition ${}^{j}s \xrightarrow{\beta} {}^{j'}s'$ of $P^{\multimap}$ appears on a pseudo-cycle, if the broadcast transition $(s, j, hasB) \xrightarrow{\beta} (s', j', betB)$ appears on a pseudo-cycle of $\tilde{P}$.

[ARZS14] uses linear programming to find such pseudo-cycles and their algorithm can be used here as well.

CHAPTER 6

Group changes

Now that the RGBN turned out to be a restriction of RMBN with a decidable PMCP for infinite runs, we want to generalize it a little bit further.
We do so by allowing processes to change between groups with rendezvous transitions. The template specifies, which rendezvous transitions allow processes to change group. Other processes active in the same transition can *pull* it to its group.
Formally, this will be done by having two group partition symbols in each rendezvous transition. When they are the same, processes taking this transition stay in their group. When they are different, a process taking this transition can change group.

6.1 Formal definition

Definition 16. The process template for rendezvous-group broadcast networks with group changes are LTS of the form $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A} \times [K] \times \mathcal{G}^2 \cup \{\beta\}, \mathcal{T} \rangle$, where $\mathcal{A}$ is set of rendezvous letters, $K \in \mathbb{N}$ and $\mathcal{G}$ is a set of group partition symbols. The group broadcast transitions are written in the from $s \xrightarrow{\beta} s'$. For each state $s \in \mathcal{S}$, there is exactly one state $s' \in \mathcal{S}$, such that $s \to s'$ is a group broadcast transition. The rendezvous transitions written in the form $s \xrightarrow[g,g']{a,k} s'$, where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $k \in \mathbb{N}$ and $G, G' \in \mathcal{G}$.

While most access functions can be taken from Definitions 1 and 9, the group partition symbols need new access functions:
$src\_grp(s \xrightarrow[g,g']{a,k} s') = g$ and $src\_grp(s \xrightarrow[g,g']{a,k} s') = g'$.
In this case, $src\_grp_a(k) = g$ and $dst\_grp_a(k) = g'$ are the unique group partition symbols associated with letter a and role k.

We require additionally that if $dst\_grp_a(k) = g'$ for letter a, a role k and a group partition symbol g' then there is a role $\hat{k}$ such that $src\_grp_a(\hat{k}) = g'$, i.e. each group partition symbol appearing as a destination symbol also appears as a source symbol in a transition with the same rendezvous letter.

As for RBGN each group partition symbol gets a group assigned in a global transition. Intuitively, when the group partition symbols in a rendezvous transition are different, a process taking that transition changes from the group assigned to the source group partition symbol to the group assigned to the destination group partition symbol.

The last requirement in Definition 16 therefore ensures that there must be another process pulling the process into its new group. Processes can not simply jump to a random other group after the initial group assignment.

Construction 9. The network $P^N = \left\langle \mathcal{S}^N \times [N]^N, \mathcal{S}_0^N \times [N]^N, \mathcal{A} \times [N]^K \cup \{\beta\} \times [N], \mathcal{T}^N \right\rangle$ is defined as for RGBN in Definition 10. Global rendezvous transitions are written in the form $(\vec{s}, \vec{l}) \xrightarrow{a,\vec{p}} (\vec{s}', \vec{l}')$, with the conditions:

$$\begin{array}{ll}
\vec{s}(n) = \vec{s}'(n) & \forall n \notin range(\vec{p}) \\
\vec{s}(\vec{p}(k)) \xrightarrow[grp_a(k)]{a,k} \vec{s}'(\vec{p}(k)) \in \mathcal{T} & \forall k \leq K \\
\vec{l}(\vec{p}(k)) = \vec{l}(\vec{p}(k')) & \forall k, k' \leq K : src\_grp_a(k) = src\_grp_a(k') \\
\vec{l}(\vec{p}(k)) = \vec{l}'(\vec{p}(k')) & \forall k, k' \leq K : src\_grp_a(k) = dst\_grp_a(k') \\
\vec{l}'(\vec{p}(k)) = \vec{l}'(\vec{p}(k')) & \forall k, k' \leq K : dst\_grp_a(k) = dst\_grp_a(k')
\end{array}$$

All results for RGBN have a analogous result for RGBN with group change. The remainder of this section will outline how the definitions, theorems and proofs from section 5 have to be adapted for RGBN with group change.

6.2 Reachability

When combining runs analogous to lemma 7, the group changes have to be considered as well. Therefore,

$$grp(\vec{\pi}_{i(r,n,m)}) \left(\sum_{\tilde{m}=1}^{m-1} dim({}_{\tilde{m}}\vec{\pi}) + n \right) = h_m(group({}_m\vec{\pi}_r)(n)) \tag{6.2.1}$$

is used instead of (5.2.1). Here, $i(r, n, m)$ refers to the index of the global transitions, where process n of run m is active for the r-th time.
I.e. compared to RGBN, the current group of each process is used, not the global one.

Algorithm 2 to decide reachability needs only minor adaptations. The main issue that needs to be mentioned is how to handle two group indicators in each transition. In iteration $q = 0$, all source and destination group partition symbols of all transitions with letter a must be the same in order to be considered for adding.

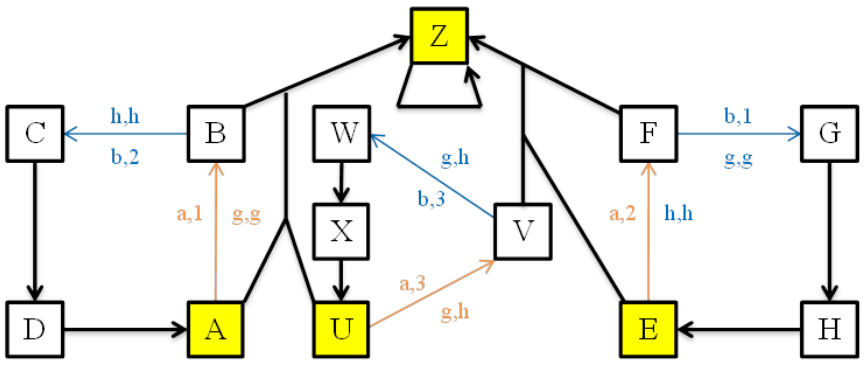

Figure 6.1: Example of a template of a rendezvous-group broadcast network with group changes.

The sets ${}_g\mathcal{S}_a$ / ${}_g\mathcal{S}'_a$ are now the set of states s for which there is a transition with source/destination s, letter a and source/destination group partition symbol g. We now go through triples $(a, g', \vec{g})$, with g' appearing on a transition with letter a as destination symbol and $\vec{g}(g') = j$, where j is the component currently calculated. It has to be the destination symbol, as states are added to the component the destination symbol is mapped to in each calculation step. If the source symbol would be used, states would be added to components already calculated in this iteration. When a triple $(a, g', \vec{g})$ is added copies of all states in ${}_g\mathcal{S}'_a$ are added to ${}^j_q\mathcal{S}$, and rendezvous transitions of the form ${}^{vecg}(g)s \xrightarrow[g,g']{a,\vec{g},k} {}^j s'$ are added to ${}^j_q\mathcal{T}$ for all transition with letter a and destination group partition symbol g'.

Theorem 5 and Lemma 8 are still valid and the same proofs apply.

Example

We will now give an example of the calculation of the unwinding template for the RGBN with group change in Figure 6.1.

Both rendezvous letters a and b describe inter-group rendezvous transitions, where one of the three roles allows a process to change group. The process taking Role 3 will change from the group the process taking Role 1 is in to the process taking Role 2 is in.

As there are no intra-group rendezvous transitions, Iteration 0 displayed in Figure 6.2 only contains broadcast transitions.

All states required to perform the global transition with letter a are present in group 1. Group partition symbol g can be mapped to Component 0, as it contains the states A and U. Group partition symbol h can be mapped to Component 1, as it contains state

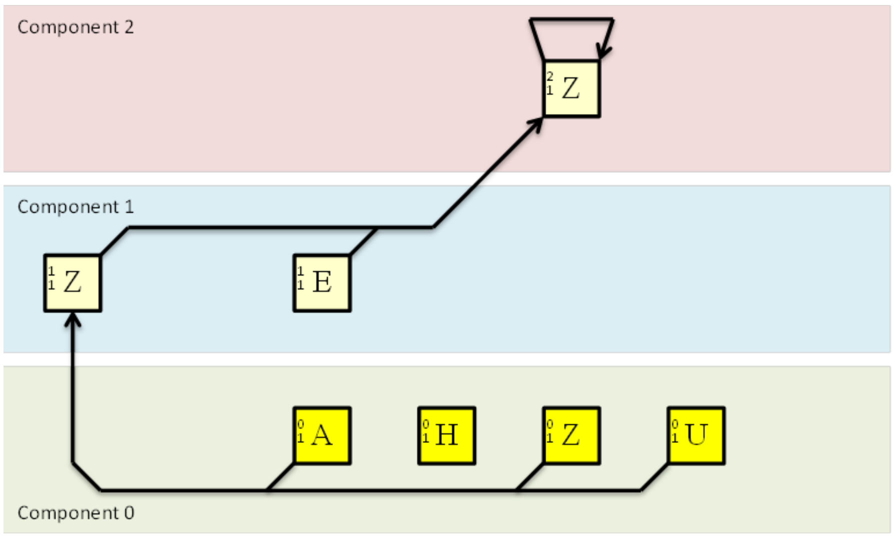

Figure 6.2: Iteration 1 of unwinding of Figure 6.1

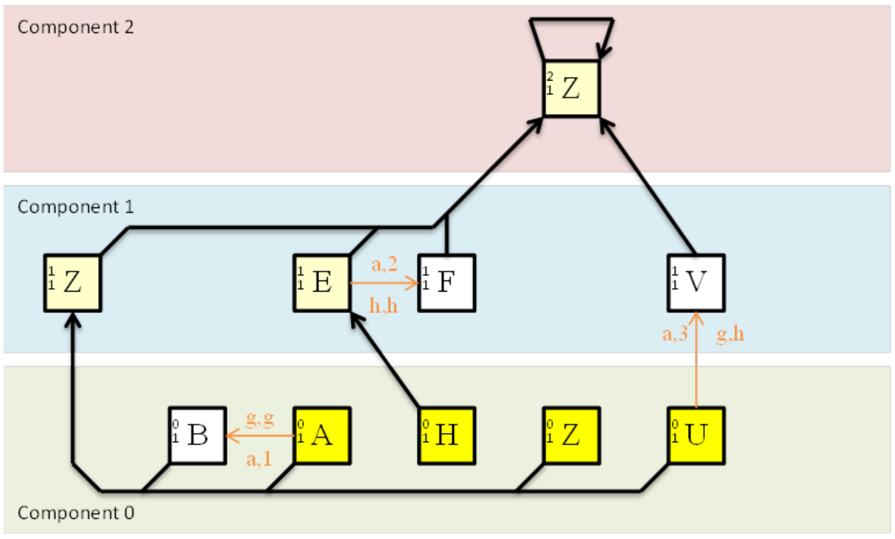

Figure 6.3: Iteration 1 of unwinding of Figure 6.1

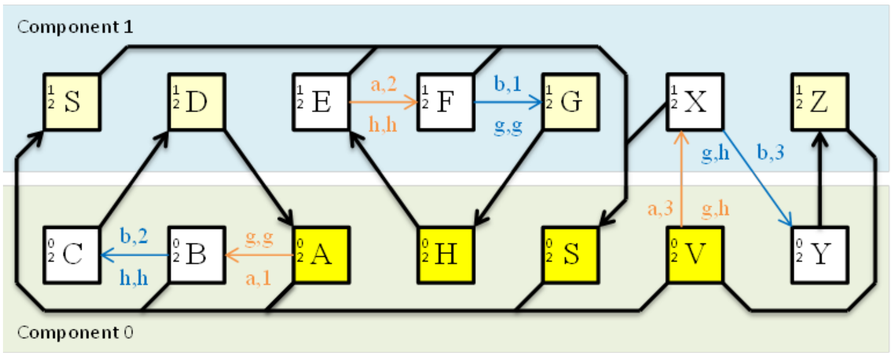

Figure 6.4: Iteration 2 of unwinding of Figure 6.1

E. The mappings are omitted from the label in Figures 6.3 and 6.4 to simplify the graphics.

In Iteration 2, the global transition b can be added as well, mapping group partition symbol g to Component 1 and h to Component 0. Component 2 now contains the same initial states as Component 0. This causes Iteration 2 to have 2 components and a cycle length of 2.
No additional states are added in the next iteration.

6.3 Infinite runs

As for RGBN without group change, the first step in analyzing infinite runs is to define colors. For broadcast transitions and non-group changing rendezvous transitions the colors are defined as for RGBN without group change.

Transition causing a process to change groups have two colors, the source color and the destination color. They depend on how often a process can leave or enter a given group using the transition.

Definition 17. Let t be a transition with $src\_grp(t) = g$ and $dst\_grp(t) = g'$ and $g \neq g'$. Then, the source/destination color of t is:

- red, if it can only occurs finitely many times in any run.
- blue, if it can occur infinitely many times with g/g' mapped to a group with finitely many group broadcasts, but only a finite number of times with g/g' mapped to a group with an infinite number of group broadcasts.
- orange, if it can occur infinitely many times with g/g' mapped to a group with an infinite number of group broadcasts, but only a finite number of times with g/g' mapped to a group with finitely many group broadcasts.

- violet, if it can occur infinitely many times with g/g' mapped to a group with an infinite number of group broadcasts and infinitely many times with g/g' mapped to a group with an infinite number of group broadcasts broadcasts.

For example, if a group-changing edge can occur infinitely many times, but can only finitely many times have its source in a group with a finite number of group broadcasts and the destination can only be a finite number of times in a group with an infinite number of group broadcasts, the edge is orange-blue.
Note that for group changing rendezvous transitions the distinction between violet and green is not necessary. The reason for this will be explained later.

As a remark, if there were multiple transitions with the same letter and role, classification of transitions would be harder. This is the main reason, why these are required to be unique in our models.

Defining the automaton

The automaton B_P^∞ is defined as for RGBN without group change for broadcast transitions and not group changing rendezvous transitions. Group changing transitions are added as follows:

Construction 10. For all group changing rendezvous transitions $t : ({}^j s \xrightarrow[G,G']{a,k} {}^j s') \in {}^o\mathcal{T}$, depending on the source and destination colors, the following transitions are added to ${}^\infty\mathcal{T}^{ini}$ and ${}^\infty\mathcal{T}^{con}$ from Construction 7:

Part	Transition	Source color	Destination color
${}^\infty\mathcal{T}^{ini}$	$(ini, s, j) \xrightarrow[\gamma:=\gamma+1]{t} (ini, s', j)$	any	any
${}^\infty\mathcal{T}^{con}$	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j)$	any	any
	$(ini, s, j) \xrightarrow[\gamma:=0]{t} (noB, s', j)$	any	any
	$(infB, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j)$	orange or violet	orange or violet
	$(noB, s, j) \xrightarrow[\gamma:=0]{t} (infB, s', j)$	blue or violet	orange or violet
	$(infB, s, j) \xrightarrow[\gamma:=0]{t} (noB, s', j)$	orange or violet	blue or violet
	$(noB, s, j) \xrightarrow[\gamma:=0]{t} (noB, s', j)$	blue or violet	blue or violet

Intuitively, adding those transitions to the B-automaton B_P^∞ has the following effect: For RGBN without group broadcast, the automaton chose either ${}^\infty\mathcal{T}^{infB}$ or ${}^\infty\mathcal{T}^{noB}$ and stayed there for the remainder of the run.
Here, each group changing rendezvous transition allows switching between ${}^\infty\mathcal{T}^{infB}$ and ${}^\infty\mathcal{T}^{noB}$. Additional, they need to guess the component the group they switch to is in.

Analogous to Theorem 6, the following result holds true for RGBN with group change.

Theorem 7. *For an RGBN template with group change P the language accepted by B_P^∞ from Construction 7 extended by Construction 10 is $exec^{inf}(P)$.*

$\supseteq$: The ability to change between ${}^\infty\mathcal{T}^{infB}$ and ${}^\infty\mathcal{T}^{noB}$ complicates the proof of this direction slightly. Let π be an execution of P and $\pi^{-\circ}$ a corresponding execution in the unwinding template $P^{-\circ}$. Let $\vec{\pi}$ be a global run, whose projection to Process 1 is $\pi^{-\circ}$. Let i_r be the index of the r-th transition, where Process 1 is active in $\vec{\pi}$. Due to the way colors are defined in Definition 11, the following must be true: There must exist an index I, such that in every group l of $\vec{\pi}$ one of the following is true:

- Group l has no group broadcast transition after index I,
  All group changing rendezvous transitions after index I are green, violet or blue and
  The source/destination color of every group changing rendezvous transition with source/destination in group l is violet or blue.
- Group l has an infinite number of group broadcast transitions,
  All group changing transitions after index I are green, violet or orange and
  The source/destination color of every group changing rendezvous transition with source/destination in group l is violet or orange.

We can therefore construct a run π^B of B_P^∞ as follows: It contains the copies of the transition $\pi^{-\circ}$ in ${}^\infty\mathcal{T}^{ini}$ until index i_R, where R are is the largest index, such that $i_R \leq I$. Then, the copy of transition $\pi_i^{-\circ}$ in ${}^\infty\mathcal{T}^{con}$ connecting to the state (d, s, j) with $d = noB$, if the group Process 1 is in after the transition has a finite number of group broadcasts and $d = infB$ otherwise. The same happens with every group changing rendezvous transition after i_r.

The counter is reset with every group changing rendezvous transition after i_R. Therefore, the counter is still bounded by $\sum_{t\in\mathcal{T}_{ov}} c_t$.

In order to prove that every run π of B_P^∞ is the projection of a global run $\vec{\pi}$ in P^∞, we are going to intuitively use twice as many copies of the witnessing pseudo-cycles arranged in twice as many groups. This gives us two sets of groups, both containing the same arrangement of copies of the witnessing pseudo-cycles as for RGBN without group change. With each group changing rendezvous transition Process 1 will change between the two sets.

As each group changing transition has two colors, there are two witnessing pseudo-cycles ${}^{src}C_t^d$ and ${}^{dst}C_t^d$:

Lemma 12. *The source/destination color of a group changing transition t is:*

- *red, iff it does not appear on any pseudo-cycle of $P^{-\circ}$;*
- *blue, iff it appears on a pseudo-cycle ${}^{y}C_t^{noB}$ of $P^{-\circ}$ with source/destination in a group without broadcast, but it does not appear on any pseudo-cycle with source/destination in a group with broadcast;*
- *orange, iff it appears on a pseudo-cycle ${}^{y}C_t^{hasB}$ of $P^{-\circ}$ with source/destination in a group with broadcast, but it does not appear on any pseudo-cycle with source/destination in a group without broadcast;*
- *violet, iff it appears on a pseudo-cycle ${}^{y}C_t^{noB}$ of $P^{-\circ}$ with source/destination in a group with broadcast and it appears on a pseudo-cycle ${}^{y}C_t^{noB}$ with source/destination in a group without broadcast;*

$y = src$ for the source color and $y = dst$ for the destination color.

The following terms have to be changed compared to RGBN without group change:

- *Job*: The jobs of the configurations of the global run $\vec{\pi}$ are now quadruples $(j, e, d, x) \in [J] \times \{hasB, betB, noB\} \times \{hasB, noB\} \times [2]$. x is called the *set* of jobs or groups.
- *Copy*: A copy of a pseudo-cycle is now a triple $({}^{y}C_t^{d}, v, x)$.
  For non-group changing transitions v is in the same range as for RGBN.
  For group changing transitions, $v = 1$ always.
- *Position*: For configurations $\vec{s}$ in $\vec{\pi}$ the position of a process n is a quadruple $({}^{y}C_t^{d}, \hat{n}, v, x)$, where $\hat{n}$ is a process in the v-th copy of the pseudo-cycle C_t^d in set $x \in [2]$.
- *Valid Position*: The position $({}^{y}C_t^{d}, \hat{n}, v, x)$ of process n, where $\hat{n}$ is in a group of kind (j, e), is valid, if it is in a group with job (j, e, d, x).
- *Reset, Flush*: Resets and flushes are now specific to a set x. A reset or flush of set x does the same a reset or flush for RGBN, but only involves the copies of set x, groups with jobs in set x and processes in positions in set x.

This now allows the construction of $\vec{\pi}$. Compared to the case of RGBN without group change, enough processes for both sets of groups are loaded initially. In the iterative construction that follows, the cases for non-group changing transitions are the same, but only involving the groups, processes, resets and flushes in the set Process 1 is in currently.

For the case of t being a group changing transition, let x be the set Process 1 is in and $\hat{x}$ the other set, where Process 1 goes to.
First, Process 1 moves into position $({}^{src}C_t^{d}, 1, 1, x)$. Then, two transitions happen, which are a combination of the first transitions of the copies $({}^{src}C_t^{d}, 1, x)$ and $({}^{dst}C_t^{d}, 1, \hat{x})$. It

allows Process 1 to change from set x to set $\hat{x}$. Let $\vec{p}^{src}$ and $\vec{p}^{dst}$ be the assignments, which processes take each role in those two first transitions of their respective cycle due to Lemma 12. The processes in the following positions take the roles k in these two transitions, which both have the letter $a = ltr(t)$:

Transition	Position	Source group partition symbol
1	$({}^{src}C^d_t, \vec{p}^{src}(k), 1, x)$	$src\_grp_a(k) \neq dst\_grp(t)$
	$({}^{dst}C^d_t, \vec{p}^{dst}(k), 1, \hat{x})$	$src\_grp_a(k) = dst\_grp(t)$
2	$({}^{dst}C^d_t, \vec{p}^{dst}(k), 1, \hat{x})$	$src\_grp_a(k) \neq dst\_grp(t)$
	$({}^{src}C^d_t, \vec{p}^{src}(k), 1, x)$	$src\_grp_a(k) = dst\_grp(t)$

All positions in swapping roles change their position to the new set. A role is swapping, if either:

- $src\_grp_a(k) \neq dst\_grp(t)$ and $src\_grp_a(k) = dst\_grp(t)$
- $src\_grp_a(k) = dst\_grp(t)$ and $src\_grp_a(k) \neq dst\_grp(t)$

All processes in a position $({}^{src}C^d_t, \vec{p}^{src}(k), 1, x)$ for a swapping role k swap their position with the process in position $({}^{dst}C^d_t, \vec{p}^{dst}(k), 1, \hat{x})$. Note that this includes by construction Process 1.

This yields a configuration isomorphic to the configuration we would have if the first transitions of $({}^{src}C^d_t, 1, x)$ and $({}^{dst}C^d_t, 1, \hat{x})$ had happened, but Process 1 is now in the set $\hat{x}$.
$({}^{src}C^d_t, 1, x)$ and $({}^{dst}C^d_t, 1, \hat{x})$ both get marked. To finish of, Process 1 goes into position for the next transition.
If t starts in a state with $infB$, a flush in set x happens. If t starts in a state with noB, a reset in set x happens instead.
Then, all groups in set x are in a basic configuration, waiting for Process 1 to come back. The groups in set $\hat{x}$ are in a valid configuration.

All process in position of pseudo-cycle ${}^{src}C^d_t$ set z and process $\vec{p}_z(k)$ for any $z \in [2]$ and a role k with $src\_grp_a(k) \neq dst\_grp(t)$ and $dst\_grp_a(k) = dst\_grp(t)$ change position to $({}^{dst}C^d_t, \vec{p}_{\hat{z}}(k), 1, \hat{z})$. This includes Process 1. by construction
All processes in position of pseudo-cycle ${}^{dst}C^d_t$ and processes $\vec{p}_z(k)$ for any $z \in [2]$ and a role k with $src\_grp_a(k) = dst\_grp(t)$ and $dst\_grp_a(k) \neq dst\_grp(t)$ change position to $({}^{src}C^d_t, \vec{p}_{\hat{z}}(k), 1, \hat{x})$.

These are all changes and additions required to prove $\subseteq$ for Theorem 7

Creating Pseudo-cycles fulfilling the conditions in Lemma 10 works with the same method as for RGBN without group change. Some details have to be observed due to processes changing groups: The Positions of the splits of the transitions of one process are determined by the broadcast of the groups. If a group broadcast is not the last broadcast,

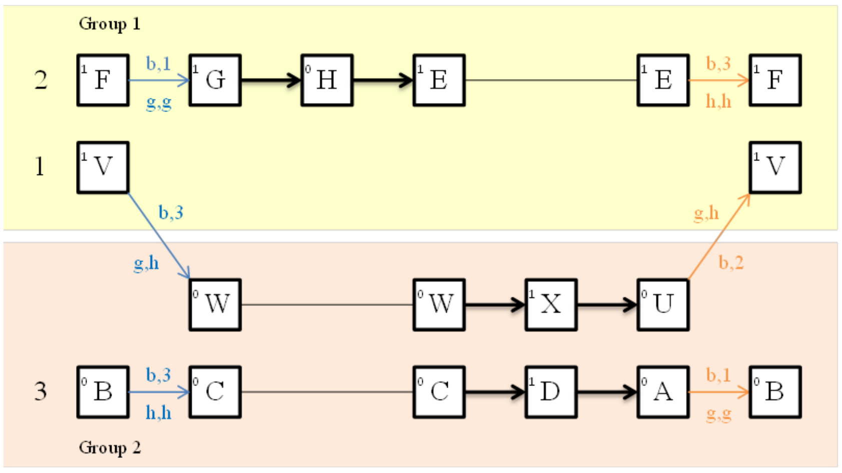

Figure 6.5: Example two transitions marked with $b, 2$

then two splits happen after it, with a new process not active in any transition.
When a group broadcast is the last broadcast of the group, there is no such additional inactive process.
The deciding factor is, whether or not the broadcast is the last broadcast in the group and not whether or not it is the last broadcast where the process is active.
The number of processes the transitions of one process are distributed among in the new pseudo-cycle is therefore $1 + b_l + 2 \cdot b_n$.
b_l is the number of group broadcasts, where the process is active and which is the last broadcast in its respective group.
b_n is the number of group broadcasts, where the process is active and which is not the last broadcast in its respective group.

This process is illustrated for the network in figure 6.1 and transition ${}^1V \xrightarrow[g,h]{b,3} {}^0W$ starting with the fixed group pseudo cycle 6.5. It has two group broadcasts in both groups, therefore each group is split up into four groups, two for each component, one with and one between broadcasts.

Figure 6.6 shows the result when combining the groups with the same job.

Colors can again be decided by an reduction to the rendezvous-broadcast network from Construction 8.
Of course, for the source color we are looking for a transition with its source in a group with the correct type. For the destination color, the type of the group of the destination of the transition is relevant.

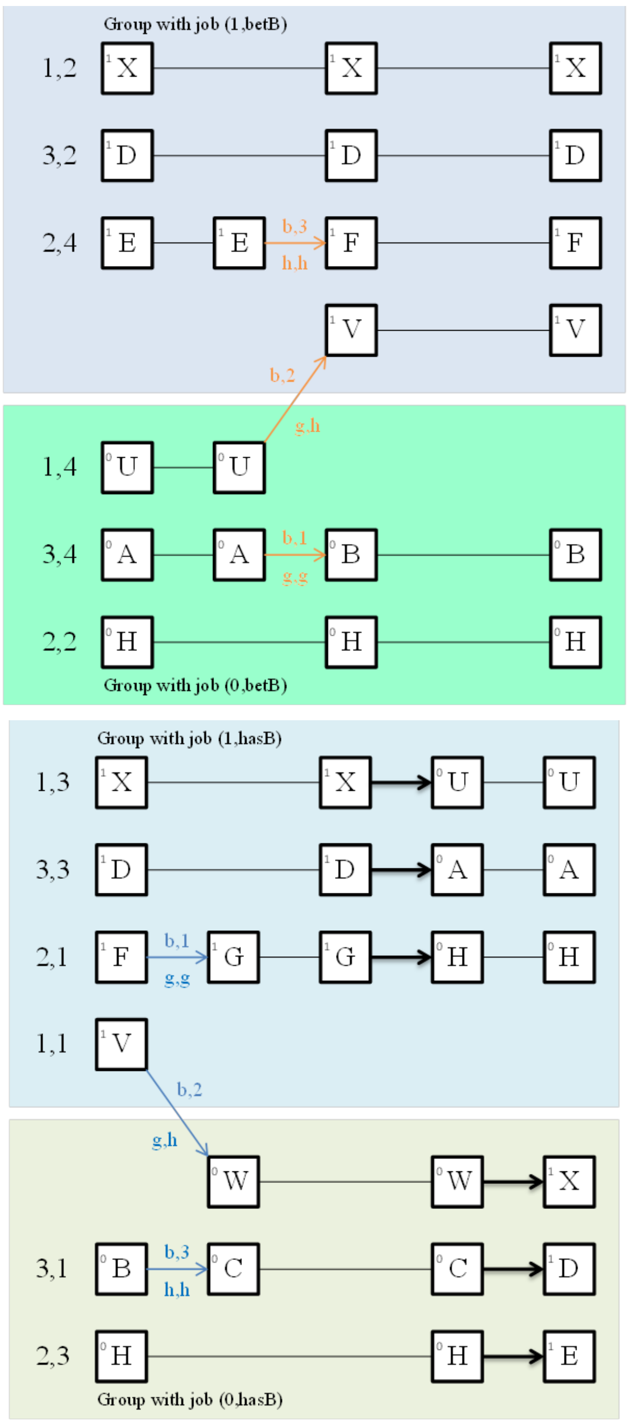

Figure 6.6: Example two transitions marked with $b, 2$

6.4 Conclusion

We set out to explore the boundary of decidability of the PMCP for rendezvous-multiple broadcast networks. We managed to refine that boundary with several decidability and undecidability results. We found decidability for finite runs and undecidability for infinite runs of the general RMBN as well as the RFGBN. We managed to give an algorithm solving the PMCP for infinite runs for the special cases of the RGBN with and without group change.

This by no means exhausts this research area. While solving the problems in this thesis, I managed to get several results not presented here. I will list them here without proof:

- Projections to several processes instead of individual processes can be described for RGBN with and without broadcast. The languages they create can be recognized by B-automaton with as many counters as there are processes in the projection.
- I also established a reverse result to [ARZS14]: For every B-automaton with one counter and no Büchi conditions, there is a rendezvous-broadcast network template, such that the language recognized by the B-automaton is an homomorphic image of the language created by the infinite executions of the template.
- The language of any B-automaton without Büchi conditions is the homomorphic image of the language created by the executions of an RMBN though.
- However, not every B-automaton without Büchi conditions and n counters can be replicated by projections of a RGBN to n processes though.

This raises the question whether there is a model of a network with a decidable PMCP for infinite runs, whose executions can correspond to any B-automaton without Büchi conditions. One avenue that might work is to allow processes in RGBN to be a member of more than one group.

Another model that may be of interest to investigate are RGBN where processes can change group during broadcast transitions. Another extension would be to introduce rendezvous transitions that only lead to empty groups.

These kind of models therefore give a lot of potential for further research.

Bibliography

[ADM04] P Aziz Abdulla, Johann Deneux, and Pritha Mahata. Multi-clock timed networks. In *Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on*, pages 345–354. IEEE, 2004.

[AJ03] Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*, 290(1):241–264, 2003.

[AKY08] Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In Franck van Breugel and Marsha Chechik, editors, *CONCUR 2008-Concurrency Theory: 19th International Conference, CONCUR 2008, Toronto, Canada, August 19-22, 2008, Proceedings*, volume 5201, pages 67–81. Springer, 2008.

[ARZS14] Benjamin Aminof, Sasha Rubin, Florian Zuleger, and Francesco Spegni. *Liveness of Parameterized Timed Networks.* Tu Vienna, Austria, 2014.

[BC06] Mikolaj Bojanczyk and Thomas Colcombet. Bounds in w-regularity. In *Logic in Computer Science, 2006 21st Annual IEEE Symposium on*, pages 285–296. IEEE, 2006.

[Boj10] Mikolaj Bojanczyk. Beynond ω-regular languages. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science - STACS 2010*, pages 11–16, Nancy,France, 2010. Proceedings of the 27th Annual Symposium on the Theoretical Aspects of Computer Science.

[CGB86] Edmund M Clarke, Orna Grumberg, and Michael C Browne. Reasoning about networks with many identical finite-state processes. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 240–248. ACM, 1986.

[EFM99] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 352–359. IEEE, 1999.

[GS92] Steven M German and A Prasad Sistla. Reasoning about systems with many processes. *Journal of the ACM (JACM)*, 39(3):675–735, 1992.

[Min67] Marvin L Minsky. *Computation: finite and infinite machines.* Prentice-Hall, Inc., 1967.

[MS97] Alexandru Mateescu and Arto Salomaa. Formal languages: an introduction and a synopsis. In *Handbook of formal languages*, pages 1–39. Springer, 1997.

CHAPTER 7

Short	Long
i.e.	that is
e.g.	for example
LTS	Linear Transition System
PMCP	Parameterized Model Checking Problem
RMBN	Rendezvous-Multiple Broadcast Network
RFGBN	Rendezvous-Fixed Group Broadcast Network
RGBN	Rendezvous-Group Broadcast Network

Table 7.1: Acronyms

Appendix

Object	Variable	Examples
Template	P	$P = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, \mathcal{T} \rangle$
State	s	$s = A; s = B; \ldots; s \in \mathcal{S}$
Initial States	$\mathcal{S}_0$	$\mathcal{S}_0 = \{A\}$
Transition	t	$t = s \xrightarrow{l} s'$;$t \in \mathcal{T}$
Source	s	$src(s \xrightarrow{l} s') = s$
Destination	s'	$dst(s \xrightarrow{a} s') = s'$
Label	a	$label(s \xrightarrow{a} s') = a$; $a \in \mathcal{A}$
Rendezvous Letter	a,b,...	$let(s \xrightarrow{a,k} s') = a$
Broadcast Letter	β,α,...	$let(s \xrightarrow{\beta} s') = \beta$
Counter	γ,δ,...	$\gamma \in \Gamma$
Counter Command	$\overline{\gamma}$	$\overline{\gamma} = (\gamma := 0)$
Role	k	$role(s \xrightarrow{a,k} s') = k$;$k \leq K$;
Group Partition Symbol	g	$grp(s \xrightarrow[g]{a,k} s') = g$;$grp_a(k) = g$
Path	π	$\pi = t_1, t_2, t_3, \ldots$
Path Index	i	π_i;π_2
Index of a Family of Paths	m	${}_m\pi$
Index of Index	r	i_r;π_{i_r}
Configuration	$\vec{s}$	$\vec{s} = (A, B, A, C)$
Process	n	$\vec{s}(n)$;$\vec{s}(3) = B$;$n \leq N$
Group Vector	$\vec{i}$	$\vec{i}(n) = 2$
Global Transition	$\vec{t}$	$\vec{t} = \vec{s} \xrightarrow[\vec{i}]{a,p} \vec{s'}$
Process Assignment	$\vec{p}$	$\vec{p} : [K] \to \mathbb{N}$; $role(\vec{s} \xrightarrow{a,\vec{p}} \vec{s'}) = \vec{p}$
Global Path	$\vec{\pi}$	$\vec{\pi}_i = \vec{t}$; $\vec{\pi}(n) = \pi$
Projection	π	$proj(\pi, n) = \pi$
Component Index	j	$(s, j) \in {}^j\mathcal{S}$; $(s, \tilde{S}) \in comp(\tilde{S})$
Calculation Iteration	q	${}_qP$