How to find algebraic semantics for temporal logics

Peter Höfner

Universität Augsburg

November 2009

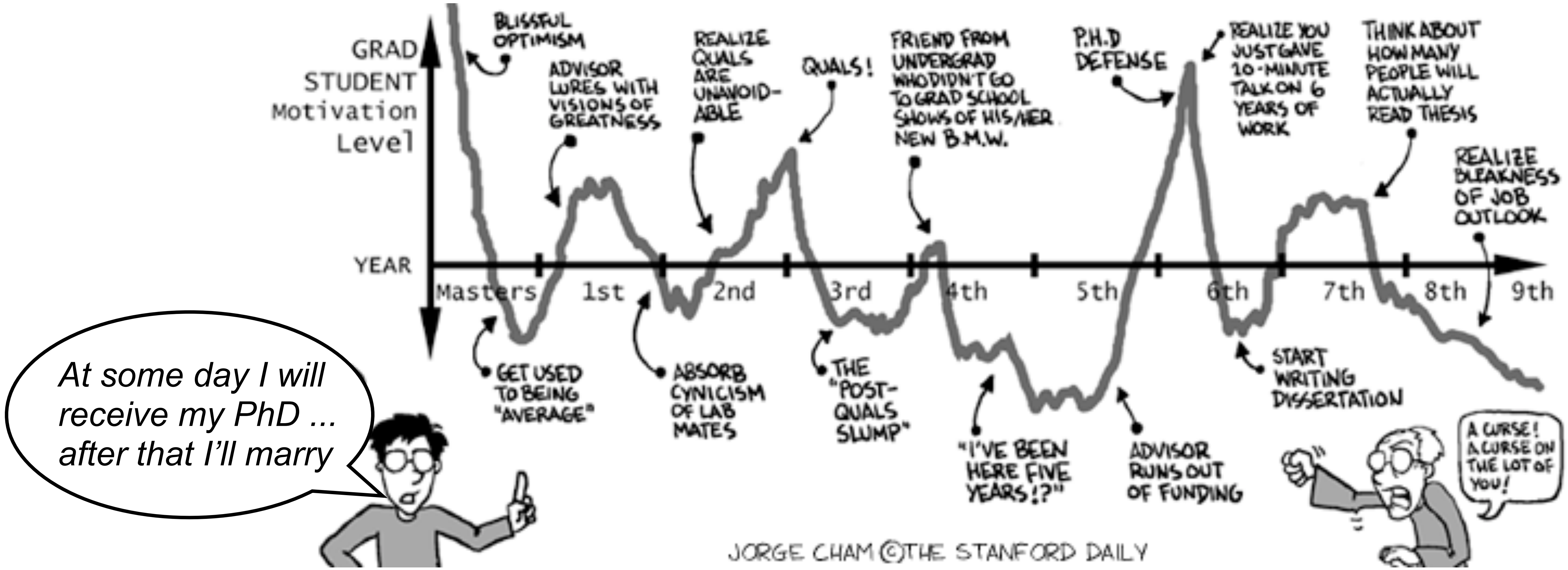
Introduction

- aims
  - short introduction to temporal logics like LTL, CTL, CTL* ...
  - derive algebraic semantics for CTL*
  - short introduction to Neighbourhood Logic
  - derive algebraic semantics for Neighbourhood Logic
  - give a general scheme how to derive algebraic semantics
- this is a *tutorial*
- 45 minutes are far too short, continue on your own

Preliminaries

- knowledge of sets, union, intersection, complementation
- some basics of propositional logic
- basic knowledge about graphs

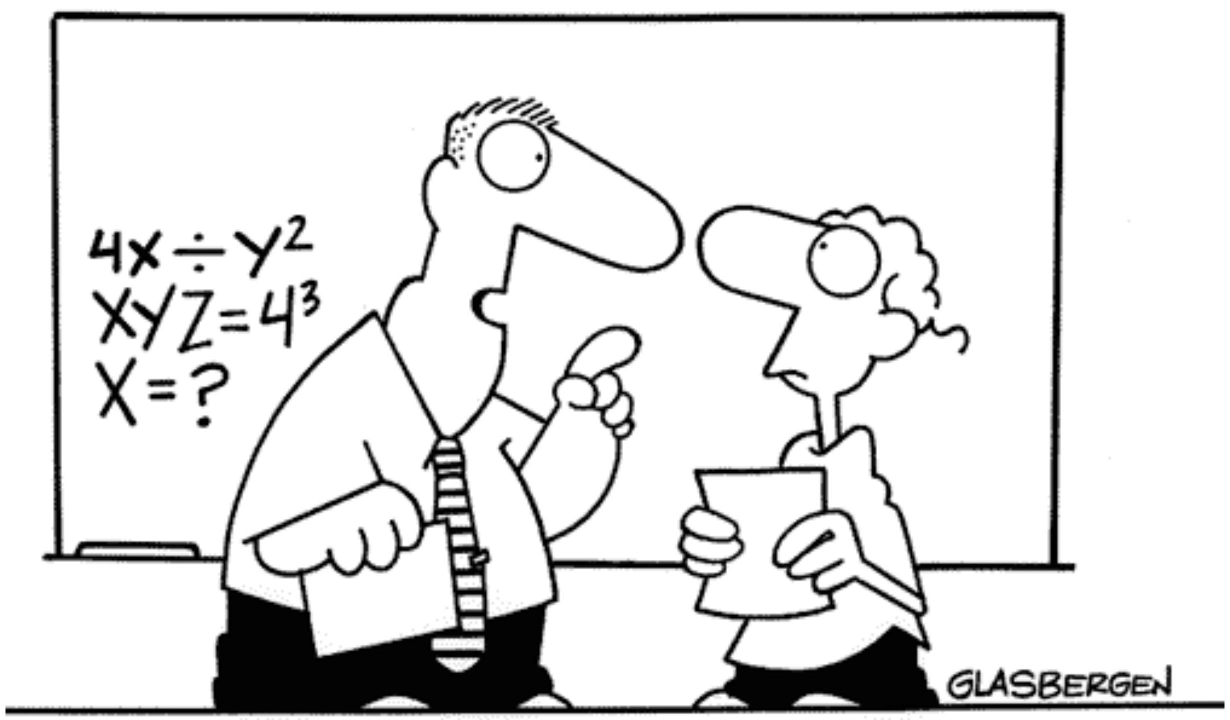
Motivation



why temporal logic

- temporal logic is everywhere
- describe temporal behaviour
- examples are: language, flow analysis, logics of programs, philosophy, etc.

Motivation



"Algebra class will be important to you later in life because there's going to be a test six weeks from now."

why algebra

- simply and concise proofs
- cross reasoning possible (unification)
  if logics are lifted to the same type of abstract algebra
- automated theorem proving with off-the-shelf software

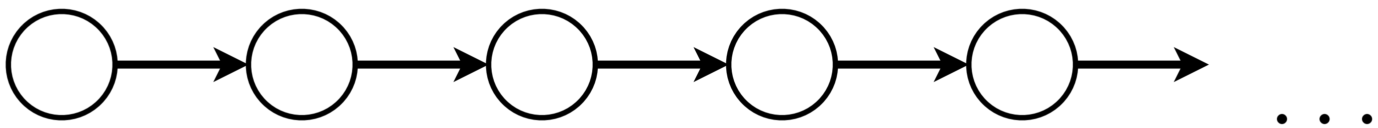
A Crash Course in Temporal Logic

- *temporal logics* describe any system of rules for representing, and reasoning about, propositions qualified in terms of time
- first studied in depth by Aristotle
- any logic which uses quantifier is a *predicate logic*;
  any logic which uses time as a sequence of states is a *temporal logic*;
- temporal logic has found an important application in formal verification
- can reason about time
- concrete temporal logics
  - linear temporal logic (LTL) [Pnueli Manna 1977]
  - computation tree logic (CTL) [Clarke Emerson 1981]
  - CTL* [Emerson Halpern 1986]
  - neighbourhood logic [Zhou Hansen 1998]
  - ...

Examples

- eventually, I will pass my PhD defense
- until I have passed my defense, I will be a grad student
- I'm always hungry
- after this and the next beer, I will only have two more beers
- as long as I stay in Doha I will not see my parents
- after program P has terminated, program Q is executed
- if each atomic program execution takes at most $5\,ms$, the whole program does not need more than $10\,s$ to terminate.
- if P terminates the next program to be executed will need variable x

LTL - Linear Temporal Logic



- time is discrete and is characterised by points
- (computation) path is a (possible infinite) sequence of states
- future is not determined (consider several paths)
- base is a finite set of atomic propositions like
  “I have a PhD”, “process 1253 is suspended”,
  “program P is executed”, etc.

LTL - Syntax

$$\Psi \;::=\; \bot \mid \Phi \mid \Psi \rightarrow \Psi \mid \mathsf{X}\,\Psi \mid \Psi\,\mathsf{U}\,\Psi$$

- Φ atomic proposition
- the first three items should be known from propositional logic
- as usual

$$\neg\varphi \;=\; \varphi \rightarrow \bot \qquad \varphi \wedge \psi \;=\; \neg(\varphi \rightarrow \neg\psi) \qquad \varphi \vee \psi \;=\; \neg\varphi \rightarrow \psi$$

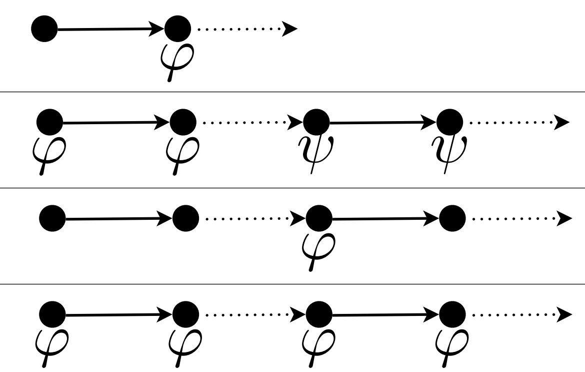
- moreover

$$\mathsf{F}\varphi \;=\; \top\,\mathsf{U}\,\varphi \qquad \mathsf{G}\,\varphi \;=\; \neg\mathsf{F}\,\neg\varphi \qquad \varphi\,\mathsf{R}\,\psi \;=\; \neg(\neg\,\mathsf{U}\,\neg\psi)$$

LTL - Semantics

LTL formulas are evaluated on paths
a state of a system satisfies an LTL formula if all paths from the given state satisfy it

$w \models \mathsf{X}\,\varphi$	ne**X**t: φ has to hold at the next state
$w \models \varphi\,\mathsf{U}\,\psi$	**U**ntil: φ has to hold (at least) until ψ
$w \models \mathsf{F}\,\varphi$	**F**inally: φ eventually has to hold
$w \models \mathsf{G}\,\varphi$	**G**lobally: φ has to hold on the entire subsequent path
$w \models \varphi\,\mathsf{R}\,\psi$	**R**elease: at first position in which φ is true, ψ ceases to be true; it is required to be true until release occurs



proper definition can be found e.g. in [Emerson 1990]

LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold
- whenever I receive an email I will send an answer
- if program P is executed once, it is executed infinitely often
- I smoked until I was 22
  (assuming that the discrete states are years)
- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor

LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold

$$\mathsf{G}\,\neg(\mathtt{started} \wedge \neg\mathtt{ready})$$

- whenever I receive an email I will send an answer

- if program P is executed once, it is executed infinitely often

- I smoked until I was 22
  (assuming that the discrete states are years)

- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor

LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold
$$\mathsf{G}\,\neg(\mathtt{started} \wedge \neg\mathtt{ready})$$
- whenever I receive an email I will send an answer
$$\mathsf{G}(\mathtt{receive} \rightarrow \mathsf{F}\,\mathtt{answer})$$
- if program P is executed once, it is executed infinitely often
- I smoked until I was 22
(assuming that the discrete states are years)
- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor

LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold
$$\mathsf{G}\,\neg(\mathtt{started} \wedge \neg\mathtt{ready})$$

- whenever I receive an email I will send an answer
$$\mathsf{G}(\mathtt{receive} \longrightarrow \mathsf{F}\,\mathtt{answer})$$

- if program P is executed once, it is executed infinitely often
$$\mathsf{G}(\neg\mathsf{P}) \vee \mathsf{GF}(\mathsf{P})$$

- I smoked until I was 22
(assuming that the discrete states are years)

- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor

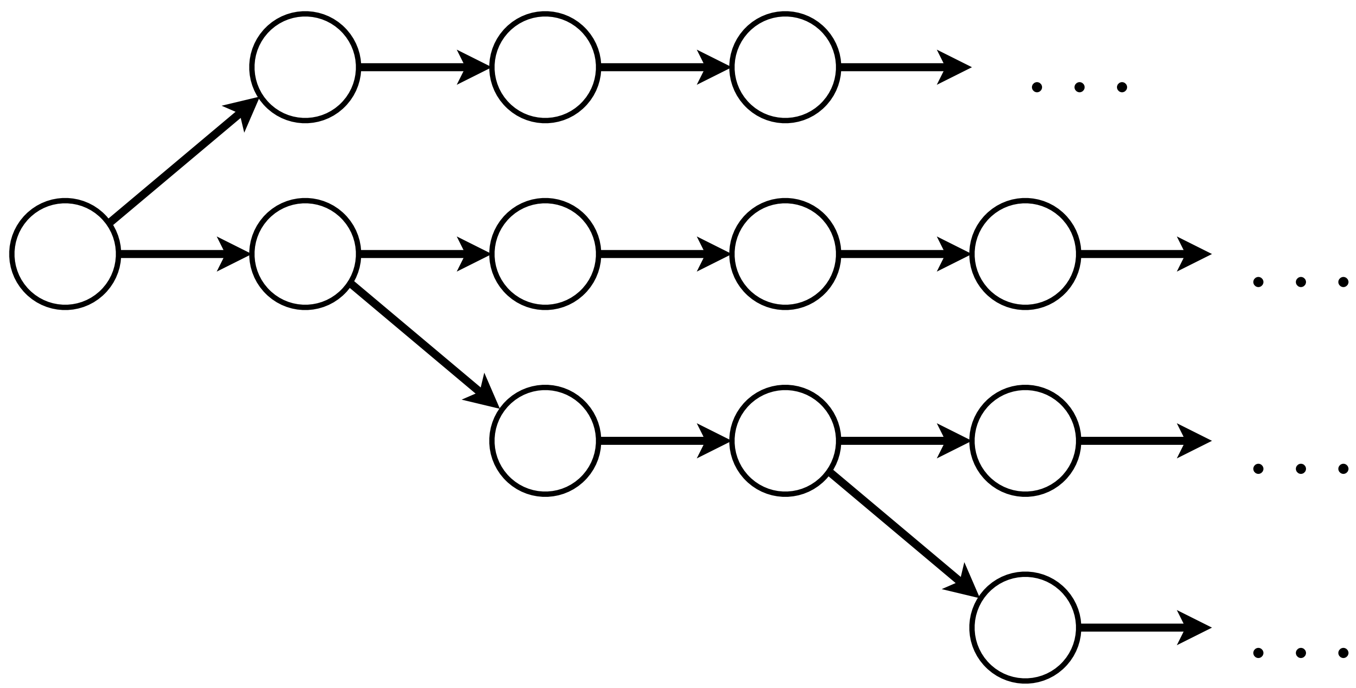
LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold
$$\mathsf{G}\,\neg(\mathtt{started} \wedge \neg\mathtt{ready})$$
- whenever I receive an email I will send an answer
$$\mathsf{G}(\mathtt{receive} \rightarrow \mathsf{F}\,\mathtt{answer})$$
- if program P is executed once, it is executed infinitely often
$$\mathsf{G}(\neg\mathsf{P}) \vee \mathsf{GF}(\mathsf{P})$$
- I smoked until I was 22
(assuming that the discrete states are years)
$$\mathtt{smoke}\,\mathsf{U}\,(\mathtt{age} = 22 \wedge \neg\mathtt{smoke})$$
- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor

LTL - Examples and Practical Patterns

- it is impossible to get to a state where $\mathtt{started}$, but $\mathtt{ready}$ does not hold
$$\mathsf{G}\,\neg(\mathtt{started} \wedge \neg\mathtt{ready})$$
- whenever I receive an email I will send an answer
$$\mathsf{G}(\mathtt{receive} \rightarrow \mathsf{F}\,\mathtt{answer})$$
- if program P is executed once, it is executed infinitely often
$$\mathsf{G}(\neg\mathsf{P}) \vee \mathsf{GF}(\mathsf{P})$$
- I smoked until I was 22 (assuming that the discrete states are years)
$$\mathtt{smoke}\,\mathsf{U}\,(\mathtt{age} = 22 \wedge \neg\mathtt{smoke})$$
- an upwards travelling escalator at the third floor does not change the direction when its passengers want to go to the fifth floor
$$\mathsf{G}(\mathtt{floor3} \wedge \mathtt{buttonpresed5} \wedge \mathtt{dirupwards} \rightarrow \mathtt{dirupwards}\,\mathsf{U}\,\mathtt{floor5})$$

CTL - Branching Time Logic



- time is discrete
- LTL cannot express existential quantifiers
- elements are now trees of states
- future is not determined
  (consider several paths of a trees or even several trees)
- base is again a finite set of atomic propositions

CTL - Syntax

the syntax of quantifies an LTL formula

$$\Psi \ ::= \ \bot \mid \Phi \mid \Psi \to \Psi \mid \mathsf{E}(\mathsf{X}\,\Psi) \mid \mathsf{E}(\Psi\,\mathsf{U}\,\Psi)$$

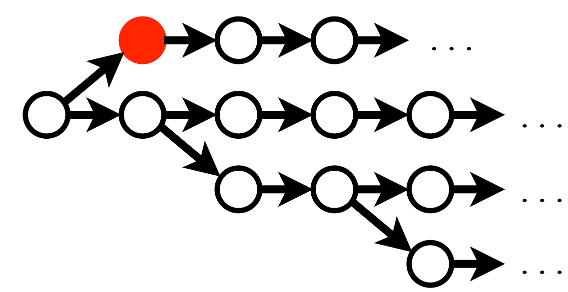
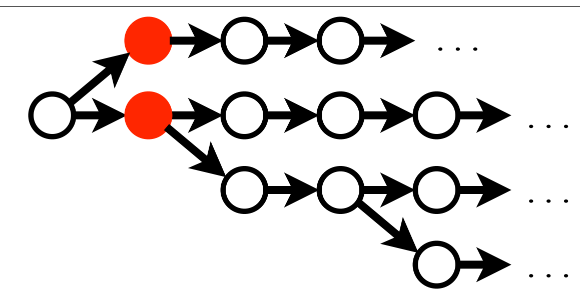
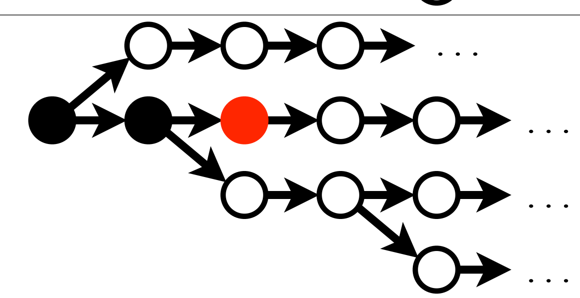
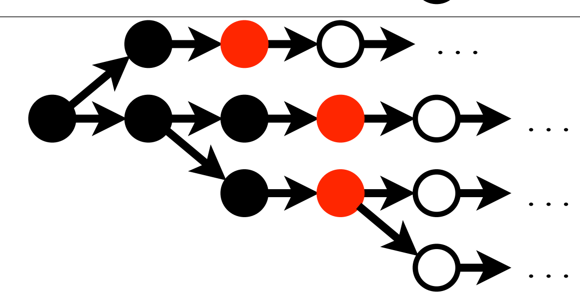
- the all quantifier is defined, as usual, via de Morgan

$$\mathsf{A}\Psi \ = \neg\mathsf{E}\neg\Psi$$

- formulas like $\mathsf{A}(\phi\,\mathsf{U}\,\psi)$ are possible by the above relations

CTL - Semantics

we only give examples, a proper definition is again in [Emerson 1990]

$t \models \mathsf{E}(\mathsf{X}\,\texttt{red})$	there is a path w that satisfies $w \models \mathsf{X}\,\texttt{red}$	
$t \models \mathsf{A}(\mathsf{X}\,\texttt{red})$	all paths w satisfy $w \models \mathsf{X}\,\texttt{red}$	
$t \models \mathsf{E}(\texttt{black}\,\mathsf{U}\,\texttt{red})$	there is a path w that satisfies $w \models \texttt{black}\,\mathsf{U}\,\texttt{red}$	
$t \models \mathsf{A}(\texttt{black}\,\mathsf{U}\,\texttt{red})$	all paths w satisfy $w \models \texttt{black}\,\mathsf{U}\,\texttt{red}$	

CTL - Examples and Practical Patterns

- whenever I receive an email I will send an answer
- if a program executes f, it can always be terminated by the user
- all paths which have a φ along them have also a ψ

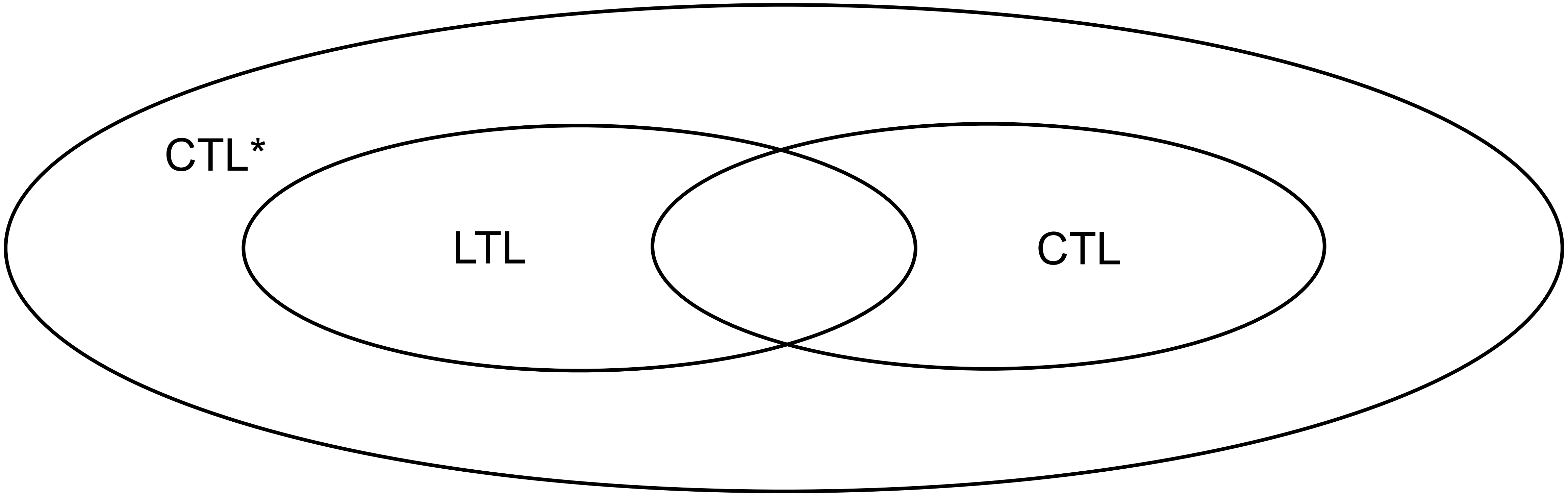
CTL - Examples and Practical Patterns

- whenever I receive an email I will send an answer
$$\mathsf{AG}(\mathtt{receive} \rightarrow \mathsf{AF}\mathtt{answer})$$
- if a program executes f, it can always be terminated by the user
- all paths which have a φ along them have also a ψ

CTL - Examples and Practical Patterns

- whenever I receive an email I will send an answer
$$\mathsf{AG}(\mathtt{receive} \rightarrow \mathsf{AF}\mathtt{answer})$$
- if a program executes f, it can always be terminated by the user
$$\mathsf{AG}(\mathtt{f} \rightarrow \mathsf{EX}\,\mathtt{terminate})$$
- all paths which have a φ along them have also a ψ

CTL*



- time is discrete
- base is again finite set of atomic propositions
- unifies paths and state formulas
- if we can derive an algebraic semantics of CTL* we have also one for CTL and LTL by restricting it to subsets

CTL* - Minimal Syntax

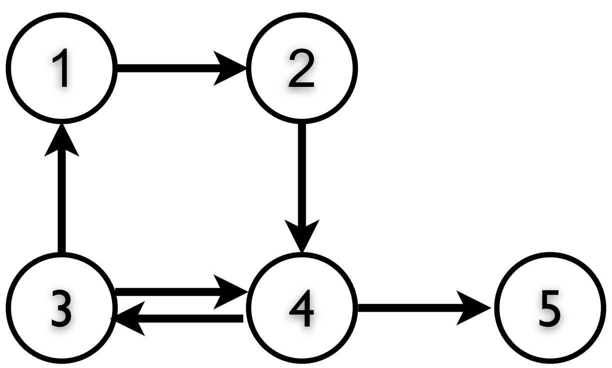
$$\begin{array}{lcl} \Sigma & ::= & \bot \mid \Phi \mid \Sigma \rightarrow \Sigma \mid \mathsf{E}\Pi, \\ \Pi & ::= & \Sigma \mid \Pi \rightarrow \Pi \mid \mathsf{X}\,\Pi \mid \Pi\,\mathsf{U}\,\Pi. \end{array}$$

- as in LTL and CTL we can define the operators
  $\wedge,\ \vee, \neg,\ \mathsf{A},\ \mathsf{G},\ \mathsf{F}$
- if we are able to give algebraic expressions for the minimal syntax we can determine the derived operators

Let’s Get Algebraic !

Graphs, Matrices and Relations

there is a close relationship between all these concepts



	1	2	3	4	6
1		x		x	
2					
3	x			x	
4			x		x
5					

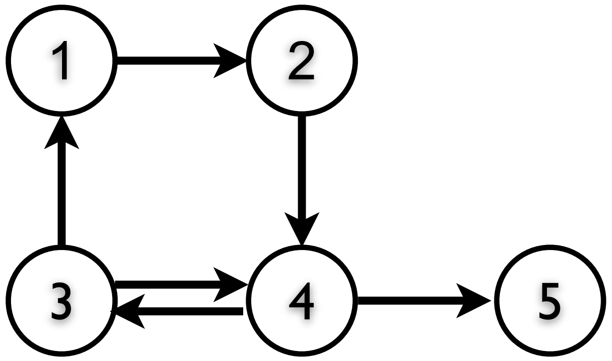
$$R = \{(1,2),\ (2,4),\ (3,1),\ (3,4),\ (4,3),\ (4,5)\}$$

graph

adjacency matrix

relation

From Relations to Paths



- relations present graphs
- composition corresponds to path fusion
- relation stores only the starting and the ending points
- the intermediate points are lost

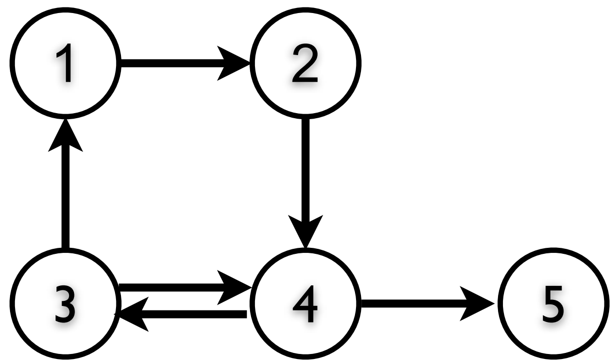
  for example there is no difference between the result of

$$(1,2)\,;(2,4) \qquad \text{and}$$
$$(1,2)\,;(2,4)\,;(4,3)\,;(3,4)$$

From Relations to Paths

- relations cannot express properties of paths or intermediate states
- use paths (sequences of nodes) and sets of paths instead
- paths con be composed if the last node of the first corresponds to the first one of the second
  $$x.s \bowtie s.z = x.s.y$$
- as in the case of relations composition of paths can be lifted to sets of paths
- moreover two sets of paths can be composed using set union
- both relations and paths form the same algebraic structure, namely quantales
  (a generalisation of relation relation algebra)

An Example for Paths



$$T = \{1.2,\ 2.4,\ 3.1,\ 3.4,\ 4.3,\ 4.5\}$$

- as in relations T^2 determines all path of length 2
- it stores all intermediate states

$$T^2 = \{1.2.4,\ 2.4.3,\ 2.4.5,\ 3.1.2,\ 3.4.3,\ 3.4.5,\ 4.3.1,\ 4.3.4\}$$

- use paths of length 1 to restrict and test elements

$$\{1\} \bowtie T^2 = \{1.2.4\}$$

From Finity to Infinity

- using the above approach one can model union and composition of finite paths
- but how to handle infinite paths?
  (important for logics)
- if an infinite path is composed with an arbitrary one, the result is the infinite path

Left Boolean Quantale

we now define the underlying algebra

- two operations: addition and composition
- addition: associative and commutative and idempotent with neutral element 0
- composition: associative, neutral element 1;
- annihilation: $0 \cdot a = 0$
- composition distributes over arbitrary sums
- the structure is also Boolean, i.e., we can define a complement satisfying the de Morgan dualities and a meet operator

Left Boolean Quantale

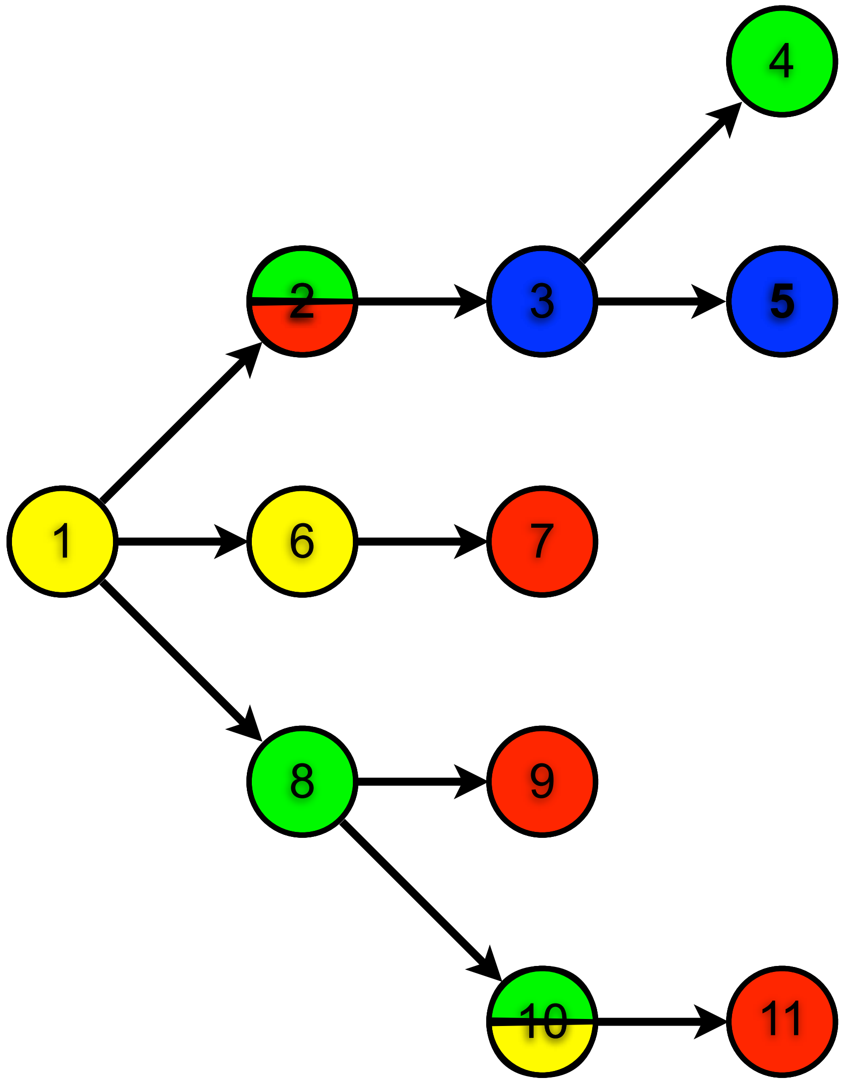
- 0 is the least element and $\top$ denotes the greatest element (union of all elements)
- finite iteration (Kleene star) is defined as the least fixpoint of $1 + a \cdot x = x$ and denoted by a^*
- infinite iteration is defined as the greatest fixedpoint of $a \cdot x = x$ and denoted by a^ω

- an example for Boolean quantales are relation algebras

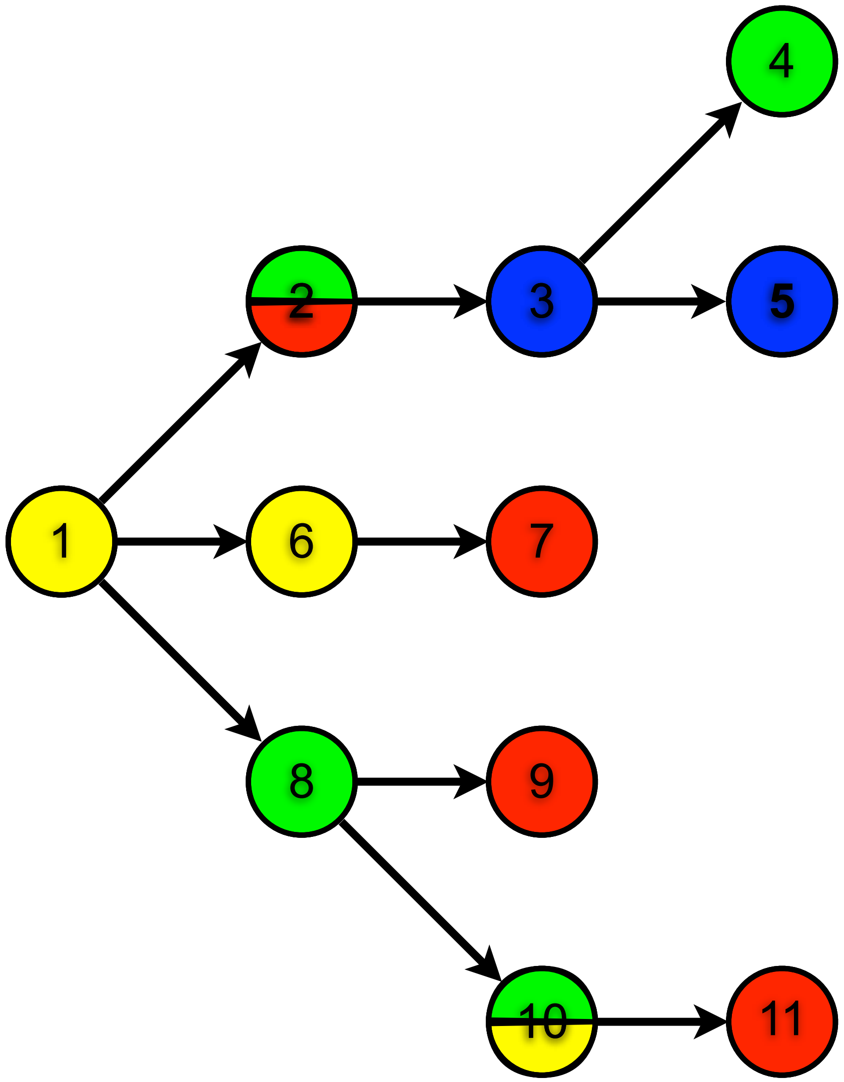
The Boolean Left Quantale of Paths

- use sets of paths as elements
- addition is set union;
  the neutral element is the empty set
- composition is defined as above;
  the neutral element is the set of all states (paths of length 1)
- finite iteration $A^* = \bigcup_{i \geq 0} A^i$
- A^ω usually contains infinite paths, however there maybe some finite paths in it

From Temporal Logics to Algebra

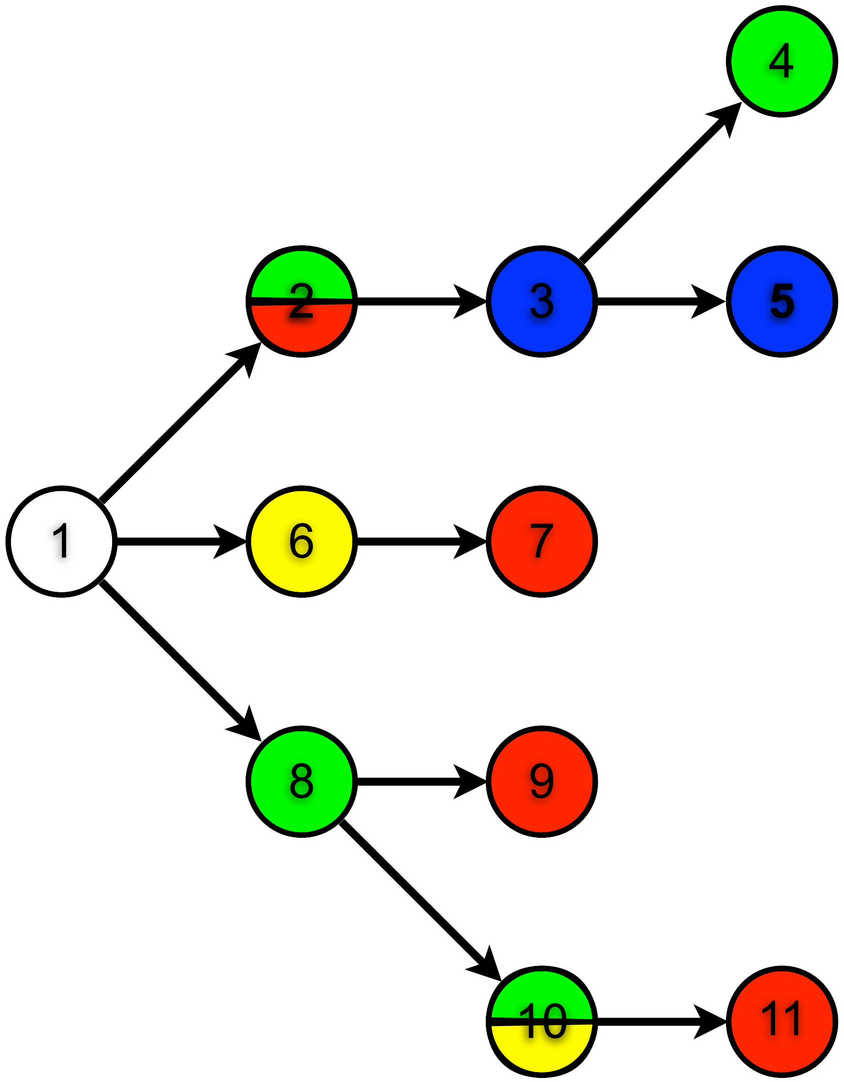


From Temporal Logics to Algebra



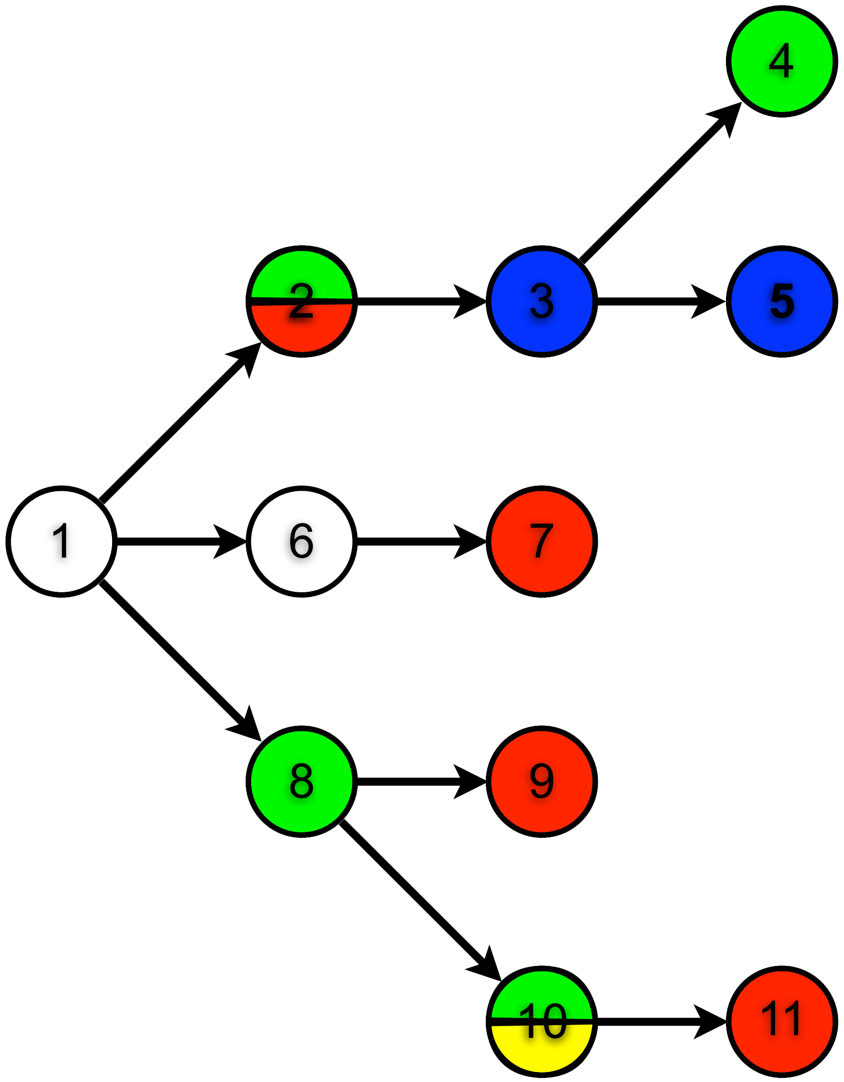
$$
\begin{aligned}
\mathrm{yel} &= \{\quad,\quad,\quad\} \\
\mathrm{gre} &= \{\quad,\quad,\quad,\quad\} \\
\mathrm{red} &= \{\quad,\quad,\quad,\quad\} \\
\mathrm{blu} &= \{\quad,\quad\}
\end{aligned}
$$

From Temporal Logics to Algebra



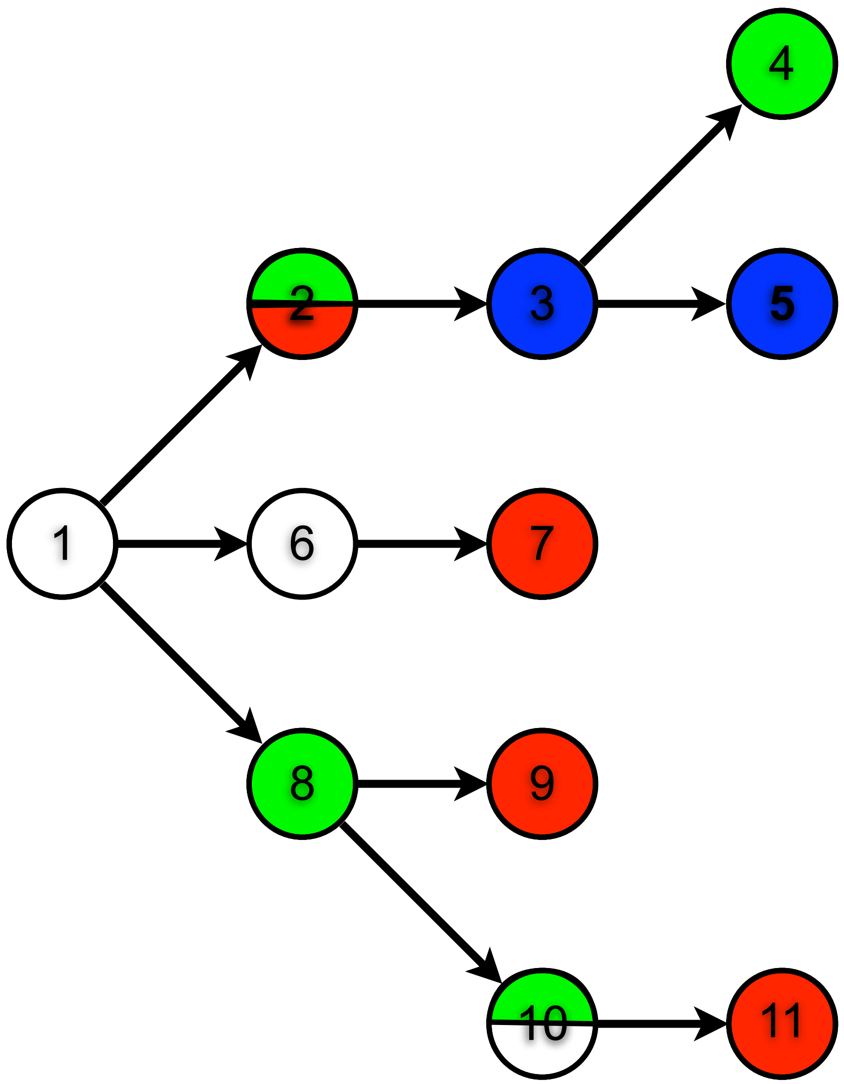
$$
\begin{aligned}
\mathrm{yel} &= \{1, \quad , \quad \} \\
\mathrm{gre} &= \{\quad , \quad , \quad , \quad \} \\
\mathrm{red} &= \{\quad , \quad , \quad , \quad \} \\
\mathrm{blu} &= \{\quad , \quad \}
\end{aligned}
$$

From Temporal Logics to Algebra



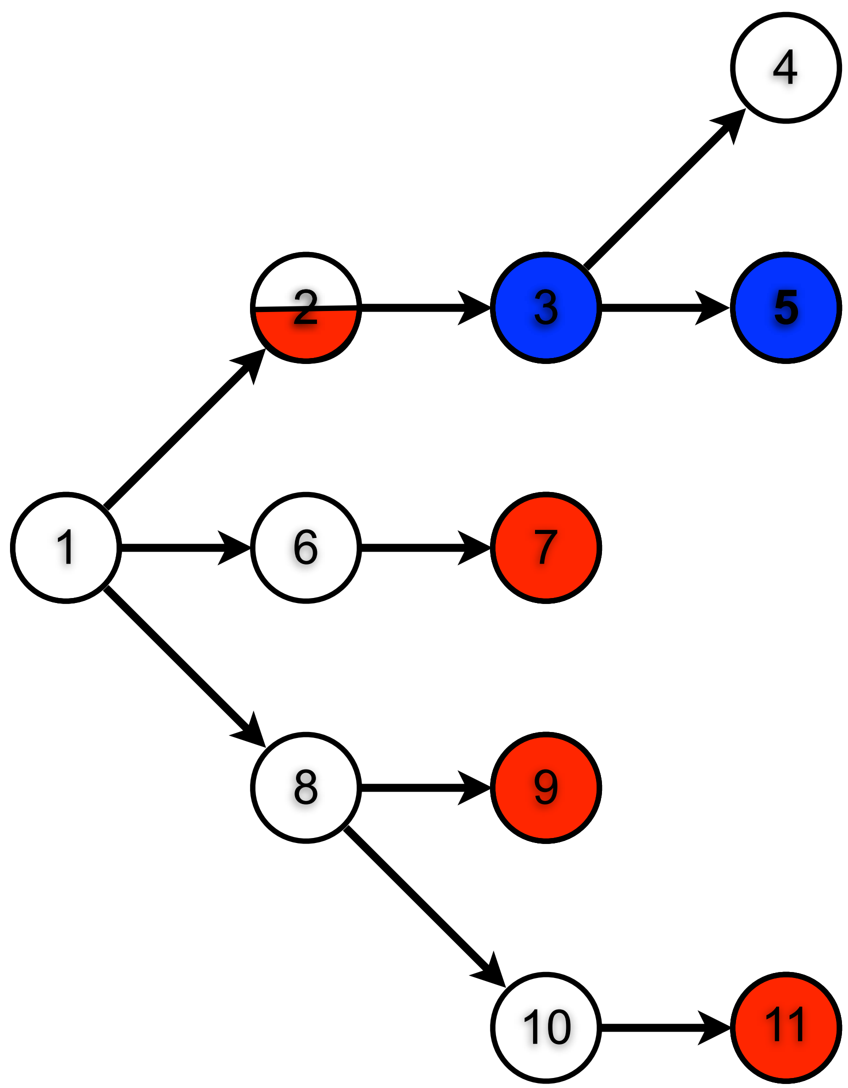
$$
\begin{aligned}
\mathrm{yel} &= \{1, 6, \quad\}\\
\mathrm{gre} &= \{\ ,\ ,\ ,\ \}\\
\mathrm{red} &= \{\ ,\ ,\ ,\ \}\\
\mathrm{blu} &= \{\ ,\ \}
\end{aligned}
$$

From Temporal Logics to Algebra



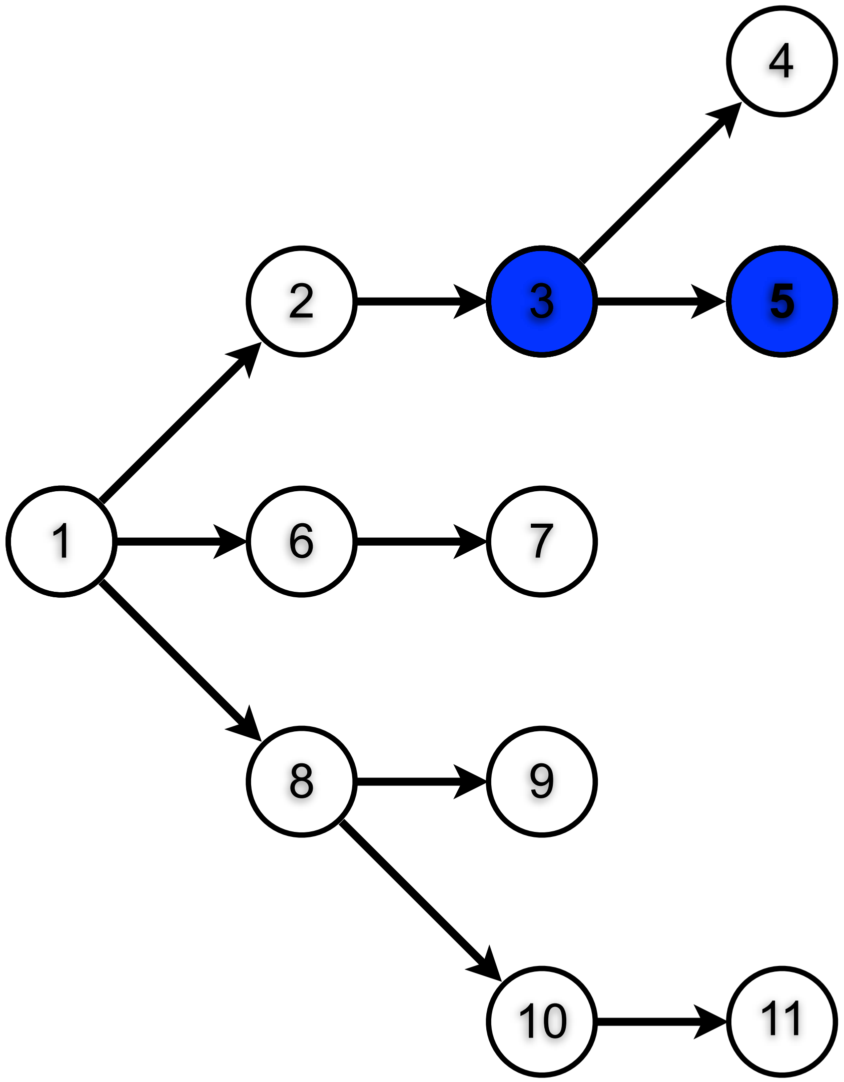
$$\begin{aligned}
\mathrm{yel} &= \{1, 6, 10\}\\
\mathrm{gre} &= \{\ ,\ ,\ ,\ \}\\
\mathrm{red} &= \{\ ,\ ,\ ,\ \}\\
\mathrm{blu} &= \{\ ,\ \}
\end{aligned}$$

From Temporal Logics to Algebra



$$\begin{aligned} \mathrm{yel} &= \{1, 6, 10\} \\ \mathrm{gre} &= \{2, 4, 8, 10\} \\ \mathrm{red} &= \{\ ,\ ,\ ,\ \} \\ \mathrm{blu} &= \{\ ,\ \} \end{aligned}$$

From Temporal Logics to Algebra



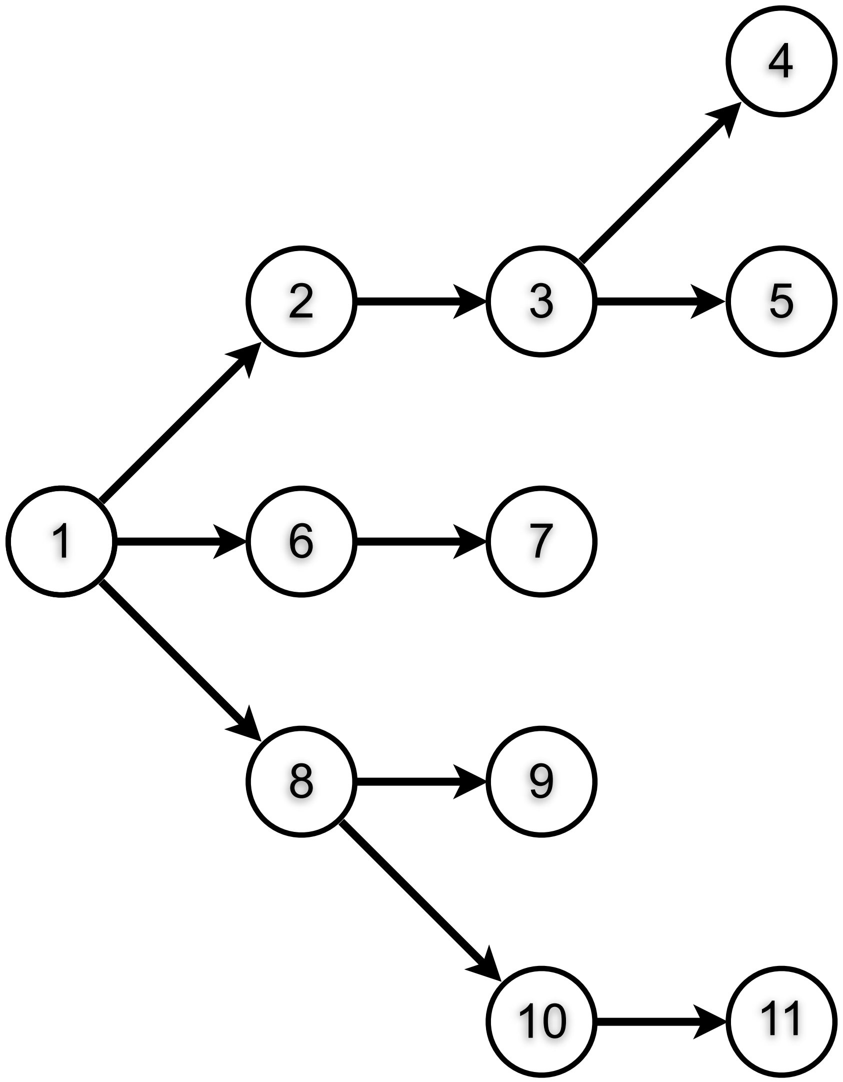
$$\mathrm{yel} = \{1, 6, 10\}$$

$$\mathrm{gre} = \{2, 4, 8, 10\}$$

$$\mathrm{red} = \{2, 7, 9, 11\}$$

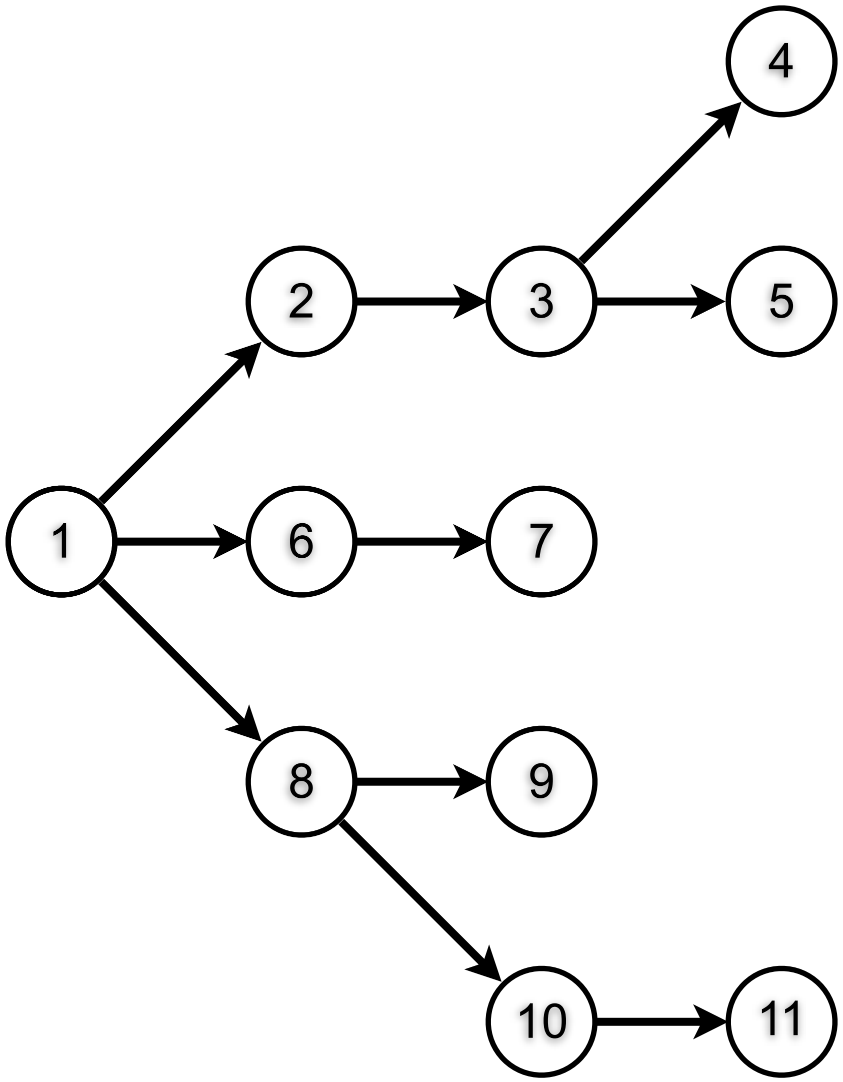
$$\mathrm{blu} = \{\ \ ,\ \ \}$$

From Temporal Logics to Algebra



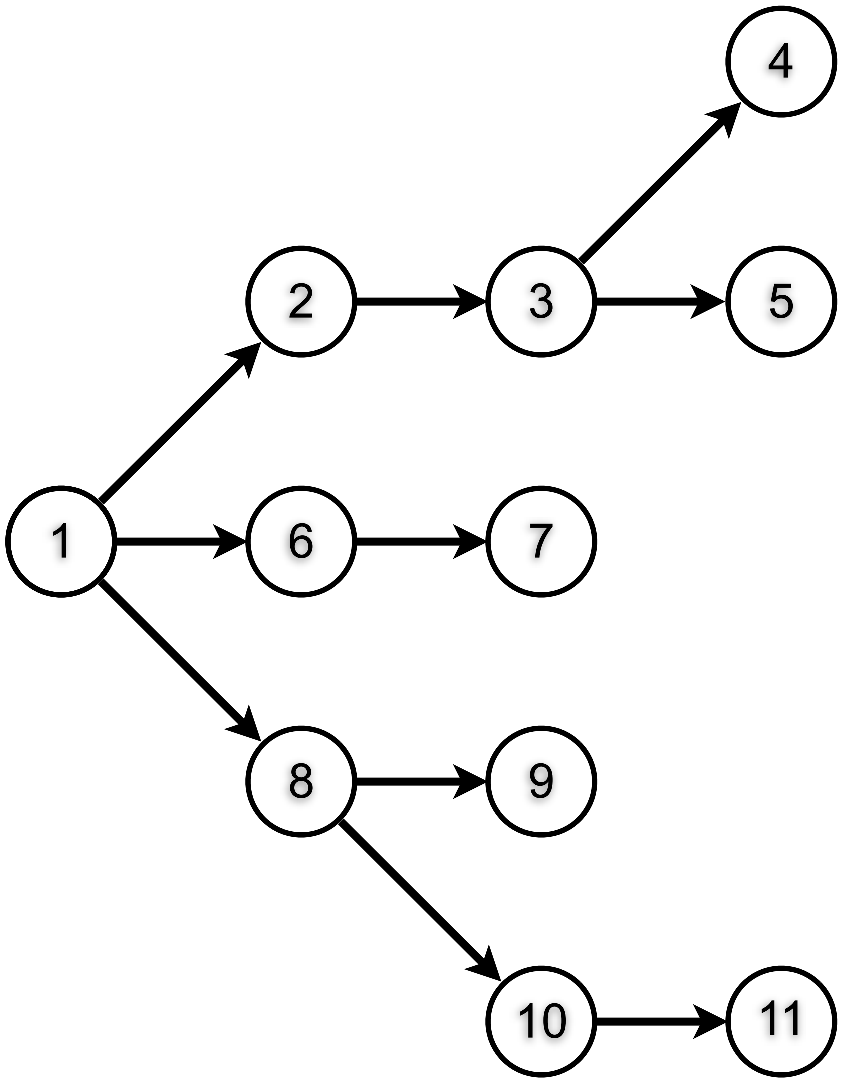
$$\mathrm{yel} = \{1, 6, 10\}$$
$$\mathrm{gre} = \{2, 4, 8, 10\}$$
$$\mathrm{red} = \{2, 7, 9, 11\}$$
$$\mathrm{blu} = \{3, 5\}$$

From Finite to Algebra



$$\begin{aligned} \text{yel} &= \{1, 6, 10\} \\ \text{gre} &= \{2, 4, 8, 10\} \\ \text{red} &= \{2, 7, 9, 11\} \\ \text{blu} &= \{3, 5\} \end{aligned}$$

From Finite to Algebra



$$\begin{aligned} \text{yel} &= \{1, 6, 10\} \\ \text{gre} &= \{2, 4, 8, 10\} \\ \text{red} &= \{2, 7, 9, 11\} \\ \text{blu} &= \{3, 5\} \end{aligned}$$

adjacent matrix as set of paths

$A = \{1.2,\ 2.3,\ \ldots\}$

Towards Algebraic Semantics

now we can characterise temporal formulas paths in the example

- all (sub)paths of the above graph that are red at the second state

- all (sub)paths that are yellow until they are green

$$\bigcup_{j \geq 0} (A^j \bowtie \texttt{gre} \cap \bigcap_{k \leq j} A^k \bowtie \texttt{yel}) \bowtie A^*$$

Towards Algebraic Semantics

now we can characterise temporal formulas paths in the example

- all (sub)paths of the above graph that are red at the second state

$$A \bowtie \mathtt{red} \bowtie A^*$$

- all (sub)paths that are yellow until they are green

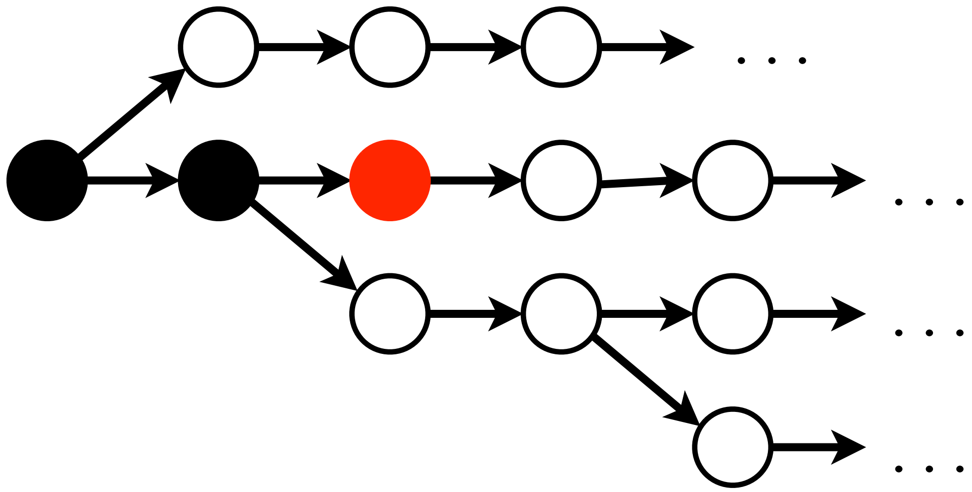
$$\bigcup_{j \geq 0} (A^j \bowtie \mathtt{gre} \cap \bigcap_{k \leq j} A^k \bowtie \mathtt{yel}) \bowtie A^*$$

Algebraic Semantics of CTL - Part I

let a be the representation of the transition system
(requirements for a will be discussed later)
concrete semantics generalises
to a function $[\![\ ]\!]$

$$
\begin{aligned}
[\![\bot]\!] &= 0, \\
[\![p]\!] &= p \cdot \top, \\
[\![\varphi \rightarrow \psi]\!] &= \overline{[\![\varphi]\!]} + [\![\psi]\!], \\
[\![\mathsf{X}\,\varphi]\!] &= a \cdot [\![\varphi]\!], \\
[\![\varphi\,\mathsf{U}\,\psi]\!] &= \bigsqcup_{j \geq 0}(a^j \cdot [\![\psi]\!] \sqcap \bigsqcap_{k<j} a^k \cdot [\![\varphi]\!]),
\end{aligned}
$$

The Existential Quantifier



to finish the algebraic semantics one has to find an algebraic expression for E

- E describes the existence of a path
- idea: determine all paths satisfying it; take the first element and continue it with an arbitrary path

Domain

- aim characterise the “first element” of an arbitrary element of an arbitrary Boolean quantale
- in relation algebra:

$$\ulcorner R = \{(t,t) \mid \exists s : (t,s) \in R\}$$

- in the algebra of paths:

$$\ulcorner T = \{(p \mid \exists x \in T . \exists s : p.s = t\}$$

- in general *domain* can be defined via a Galois connection

$$\ulcorner a \leq p \Leftrightarrow a \leq p \cdot \top$$

where p is an element of the maximal Boolean subset below 1

Algebraic Semantics of CTL - Part II

let a be the representation of the transition system (requirements for a will be discussed later)
concrete semantics generalises
to a function $[\![\ ]\!]$

$$
\begin{array}{rcl}
[\![\bot]\!] & = & 0, \\
[\![p]\!] & = & p \cdot \top, \\
[\![\varphi \rightarrow \psi]\!] & = & \overline{[\![\varphi]\!]} + [\![\psi]\!], \\
[\![\mathsf{X}\,\varphi]\!] & = & a \cdot [\![\varphi]\!], \\
[\![\varphi\,\mathsf{U}\,\psi]\!] & = & \bigsqcup_{j \geq 0} (a^j \cdot [\![\psi]\!] \sqcap \bigsqcap_{k<j} a^k \cdot [\![\varphi]\!]), \\
[\![\mathsf{E}\varphi]\!] & = & \ulcorner [\![\varphi]\!] \cdot \top
\end{array}
$$

Properties

- we can now determine the derived operators. e.g.,

$$[\![\mathsf{F}\varphi]\!] = a^* \cdot [\![\varphi]\!]$$

- more properties and longer discussions can be found in [Möller Höfner Struth 06]
- since $\mathsf{X}\neg\varphi \Leftrightarrow \neg\mathsf{X}\varphi$ the underlying transition a has to satisfy

$$\forall b : \overline{a \cdot b} = a \cdot \overline{b}$$

- from that it is easy to derive semantics for CTL and LTL (for more details see [Möller Höfner Struth 06])

An Advantage of Algebra

- use of off-the-shelf automated theorem provers
- problem: quantales are higher-order structures; at the moment theroem provers are only really good for first-order structures
- but one can use first-order logics to show parts of the properties
- today, Dang shows how to encode quantales for higher-order theorem provers

- Let’s do a toy example: Show that
$$[\![\mathrm{EXE}\varphi]\!] = [\![\mathrm{EX}\varphi]\!]$$

How We Derived the Semantics

- instead of looking at single states and paths (trees), we worked with **sets of states and paths**
- abstract from concrete operations like set union and set complement to abstract one
- most often there will be operations for choice, composition and (in)finite iteration
- if this is the case quantales seems to be one of the best abstract algebras

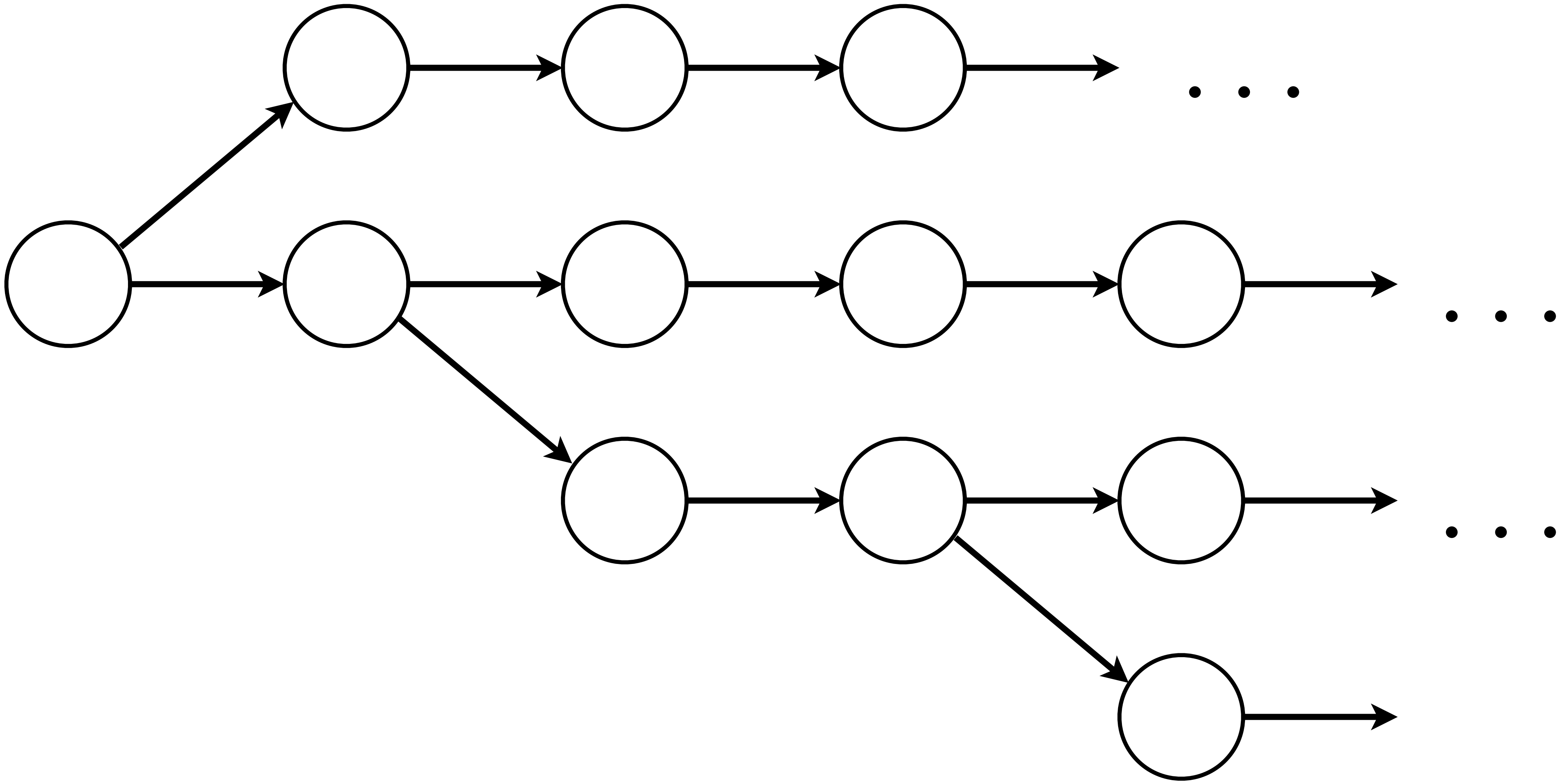
Points versus Intervals
Discrete versus Continuous

- LTL, CTL and CTL* are based points in time
- most temporal formalisms developed for program reasoning do so
- however often intervals seem to be more realistic; especially in the context of realtime systems
- if logics are point-based, time has to be discrete
- if logics are interval-based, time may be continuous

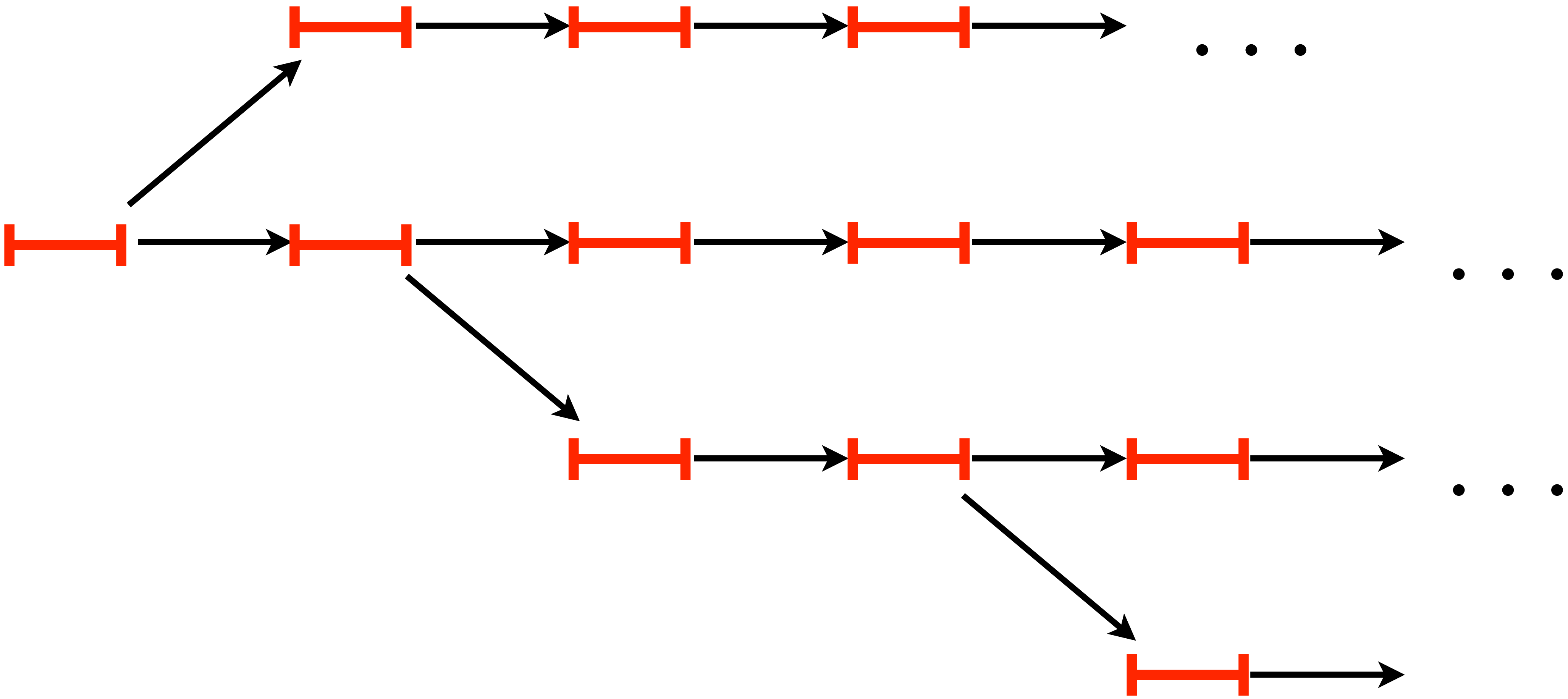
NL - Neighbourhood Logic

- a “universal” interval-based logic
- developed by Zhou and Hansen [Zhou Hansen 1996]
- subsumes at least 10 other interval logics such as
  - interval temporal logic (ITL) [Halpern Manna Moszkowski 1983]
  - interval logic (IL) [Dutere 1995]
  - Allen’s interval logic [Allen1983]
  - Venema’s chopping logic [Venema 1991]
- interval-based; hence it allows continuous time
- used for the analysis of real-time and hybrid systems

From CTL to NL (informal)



From CTL to NL (informal)



NL - Syntax

- the syntax includes temporal, functional variables etc which we skip here for simplicity
- moreover we skip a detailed discussion of functions evaluating intervals
- the remaining language of NL formulas is defined by

$$\Phi \ ::= \ \Phi \wedge \Phi \mid \neg\Phi \mid (\exists x)\Phi \mid \Diamond_l \Phi \mid \Diamond_r \Phi$$

- skipped details can be found in [Zhou Hansen 2004]
- from this minimal syntax one can again derive further operators like

$$(\forall x)\varphi = \neg(\exists x)\neg\varphi, \ \Box_l \varphi = \neg\Diamond_l\neg\varphi, \ \Box_r\varphi = \neg\Diamond_r\neg\varphi$$

NL - Semantics

$$\overbrace{u \longrightarrow y}^{\varphi}\overbrace{\longrightarrow z}^{\Diamond_l \varphi} \qquad\qquad \overbrace{y \longrightarrow z}^{\Diamond_r \varphi}\overbrace{\longrightarrow v}^{\varphi}$$

where $u = y - \delta$ where $v = z + \delta$

$$
\begin{array}{lll}
[y,z] \models_{\mathcal{J},\mathcal{V}} \neg\varphi & \text{iff} & [y,z] \not\models_{\mathcal{J},\mathcal{V}} \varphi, \\
[y,z] \models_{\mathcal{J},\mathcal{V}} \varphi \vee \psi & \text{iff} & [y,z] \models_{\mathcal{J},\mathcal{V}} \varphi \ \text{ or } \ [y,z] \models_{\mathcal{J},\mathcal{V}} \psi, \\
[y,z] \models_{\mathcal{J},\mathcal{V}} (\exists x)\varphi & \text{iff} & [y,z] \models_{\mathcal{J},\mathcal{V}'} \varphi \text{ for some } \mathcal{V}' \text{ that agrees with } \mathcal{V} \\
 & & \text{for all global variables } u \neq x \\
[y,z] \models_{\mathcal{J},\mathcal{V}} \Diamond_l \varphi & \text{iff} & \exists \delta \geq 0 : [y-\delta, y] \models_{\mathcal{J},\mathcal{V}} \varphi \\
[y,z] \models_{\mathcal{J},\mathcal{V}} \Diamond_r \varphi & \text{iff} & \exists \delta \geq 0 : [z, z+\delta] \models_{\mathcal{J},\mathcal{V}} \varphi
\end{array}
$$

NL - Getting Algebraic

- as before we do not use single elements
- that means instead looking at a single interval satisfying φ we look at a set

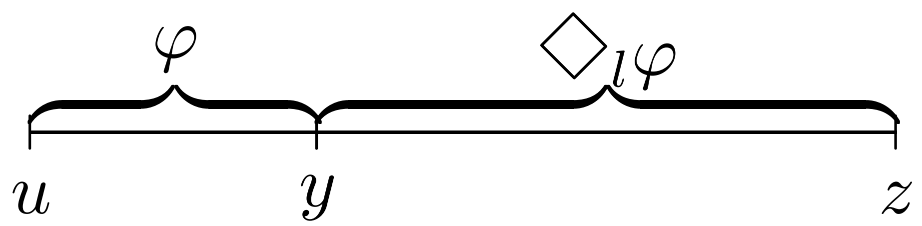
$$\mathbb{I}_\varphi =_{df} \{[y,z] : [y,z] \models \varphi\}$$

- similar to paths we define operations on sets of intervals
  - union as addition
  - point-wise lifted interval composition as multiplication
  - complement on sets
- this immedeately yields again a Boolean (left) quantale with

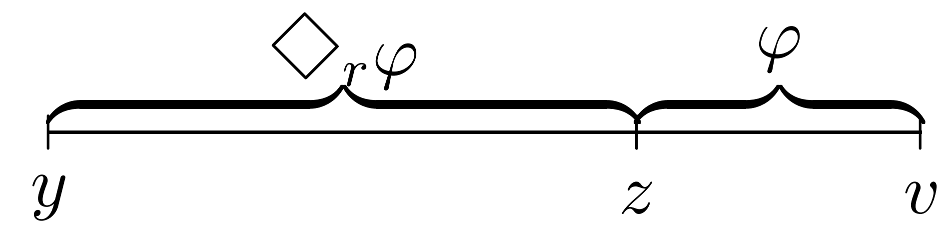
$$\ulcorner I = \{[x,x] \mid \exists y : [x,y] \in I\}$$

- one may add a right-open intervals $[x, \infty)$

Algebraic Semantics for NL (a snapshot)



where $u = y - \delta$

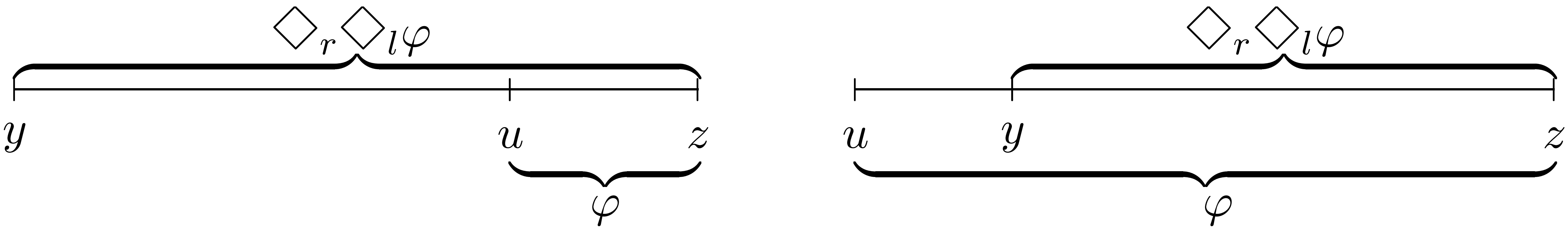


where $v = z + \delta$

$$[\![\Diamond_l \varphi]\!] = [\![\varphi]\!]^{\urcorner} \cdot \top$$
$$[\![\Diamond_r \varphi]\!] = \top \cdot {}^{\ulcorner}[\![\varphi]\!]$$

where codomain $^{\urcorner}$ is defined symmetrically to domain

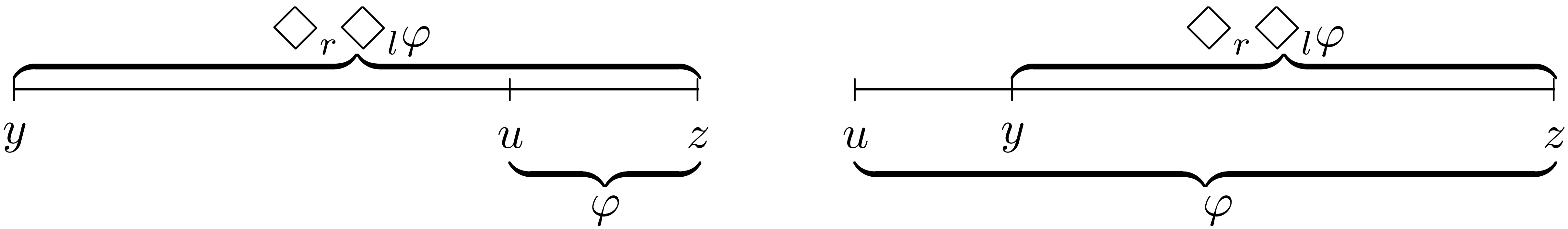
Algebraic Semantics for NL (a snapshot)



$$[\![\Diamond_r \Diamond_l \varphi]\!] = \top \cdot [\![\varphi]\!]^{\urcorner}$$
$$[\![\Diamond_l \Diamond_r \varphi]\!] = {}^{\ulcorner}[\![\varphi]\!] \cdot \top$$

(the full algebraic semantics as well as a lot of properties can be found in [Höfner Möller 2008])

Algebraic Semantics for NL (a snapshot)



$$[\![\Diamond_r \Diamond_l \varphi]\!] = \top \cdot [\![\varphi]\!]^{\urcorner}$$
$$[\![\Diamond_l \Diamond_r \varphi]\!] = {}^{\ulcorner}[\![\varphi]\!] \cdot \top$$

from an algebraic point of view this corresponds to $[\![\mathsf{E}\varphi]\!]$

(the full algebraic semantics as well as a lot of properties can be found in [Höfner Möller 2008])

CTL* vs. NL

- all presented logics can be algebraically characterised by quantales
- the resulting formulas coincide to some extent
- this shows a close relationship between the logics and allows cross-reasoning
- this was not known before the algebraization

How to find algebraic semantics for temporal logics

- instead of looking at single elements, work with **sets**
- abstract from concrete operations like set union and set complement to abstract one
- most often there will be operations for choice, composition and (in)finite iteration
- if this is the case quantales seems to be one of the best abstract algebras

Modal Logics

- like for temporal logics there are ways for modal logics
- Möller started to look at these logics [Möller 2008]
- there seem to be the same schemata lying behind
- there is a lot of more work to do

References and Further Reading

computation tree logic (CTL*)

[Huth Ryan 2004] *Logic in Computer Science - Modelling about Systems*. Cambridge University Press

[Emerson 1990] *Temporal and Modal Logic.* In J. van Leeuwen (ed.), Handbook of Theoretical Computer Science

[Gabbay Hodkinson Reynolds 1994] *Temporal Logic (Volume 1): Mathematical Foundations and Computational Aspects.* Oxford University Press

[Clark Grumberg Peled 1999] *Model Checking.* MIT Press

[Emerson Halpern 1985] *Decision procedures and expressiveness in the temporal logic of branching time.* Journal of Computer and System Sciences 30 (1): 1–24

neighbourhood logic (NL)

[Zhou Hansen 1998] *An Adequate First Order Interval Logic.* In W. de Roever, H. Langmaack and A. Pnueli (eds.), Compositionality: The Significant Difference International Symposium, LNCS 1536, pages 584-608, Springer

[Roy Zhou 1997] *Notes on Neighbourhood Logic.* Technical report 97, The United Nations University UNU/IIST

[Zhou VanHung Xiaoshan 1995] *A Duration Calculus with Infinite Intervals.* In Fundamentals of Computation Theory, 16-41, Springer

[Zhou Hansen 2004] *Duration Calculus: A Formal Approach to Real-Time Systems*, Springer

References and Further Reading

more logics

[Halpern Manna Moszkowski 1983] *A Hardware Semantics Based on Temporal Logics.* In J. Diaz (ed.), Automata, Languages and Programming, LNCS 154, pages 278-291, Springer

[Dutere 1995] *Complete Proof System for First-order Interval Temporal Logic.* Symposium on Ligic in Computer Science, pages 36-42, IEEE

[Allen 1983] *Maintaining Knowledge about Temporal Intrvals.*

[Venema 1991] *A Modal Logic for Chopping Intervals.* Logic and Computation 1(4), pages 453-547.

temporal logics and algebraic semantics

[Möller Höfner Struth] *Quantales and Temporal Logics.* In M. Johnson, V. Vene (eds.), Algebraic Methodology and Software Technology, LNCS 4019, pages 263-277, Springer

[Höfner Möller 2008] *Algebraic Neighbourhood Logic.* Algebraic and Logic Programming 76, pages 35-59

[Möller2009] *Knowledge and Games in Modal Semirings.* In R. Berghammer, B. Möller, G. Struth (eds.) Relations and Kleene Algebra in Computer Science, LNCS 4988, pages 320-336, Springer

References and Further Reading

semirings and quantales

[Hebisch Weinert 1998] *Semirings - Algebraic Theory and Applications in Computer Science.* World Scientific

[Möller 2007] *Kleene getting lazy.* Science of Computer Programming 65, pages 195-214.

[Höfner 2009] *Algebraic Calculi of Hybrid Systems, PhD Thesis. University of Augsburg*

algebraic domain

[Desharnais Möller Struth 2006] *Kleene Algebra with Domain.* ACM Transaction on Computational Logic 7(4), pages 798-833

[Desharnais Jipsen Struth 2009] *Domain and Antidomain Semigroups.* In R. Berghammer, A. Jaoua, B. Möller (eds.) Relations and Kleene Algebra in Computer Science, LNCS 5827, pages 73-87, Springer

automation in algebra

[Höfner Struth 2007] *Automated Reasoning in Kleene Algebra.* In F. Pfennig (ed.), Automated Deduction, LNAI 4603, pages 279-294, Springer

[Höfner Struth 2008] *On Automating the Calculaus of Relations.* In A. Armando, P. Baumgartner, G. Dowak (eds.) Automated Reasoning, LNAI 5159, pages 50-66, Springer

[Dang 2009] *Automated Higher-Order Reasoning in Quantales.* PhDProgramme, RelMiCS

[Höfner 2006] *Database for Automated Proofs.* http://www.kleenealgebra.de