Temporal logic

Dr. Constantinos Constantinides, P.Eng.

Department of Computer Science and Software Engineering
Concordia University Montreal, Canada

December 31, 2021

Propositional vs. modal logic

- In classical (propositional) logic, the meaning and the truthfulness of a formula are both constant.
- For example, some propositions such as *"the earth is round"* are true all the time, whereas some other such as *"the moon is made of cheese"* are false all the time.

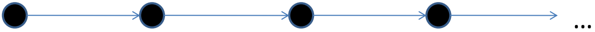
Introduction to temporal logic

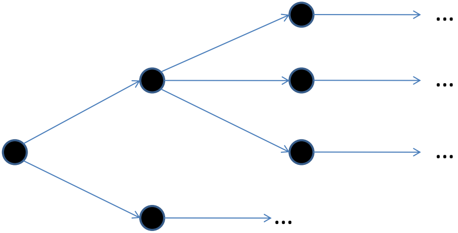
- *Modal logic* is a family of related systems that provide extensions to formal logic.
- A *modal* is an expression that is used to qualify the truthfulness of a statement.
- Of particular interest to Computer Science is an area of modal logic called *temporal logic*.
- As the name suggests, temporal logic provides concepts related to time as extensions to formal logic.
- In temporal logic, the truthfulness property of a formula can vary, i.e. the formula can be true sometimes, and it can be false sometimes but cannot simultaneous be both true and false at any time.

Linear vs. branching models of time

- We view time as a set of moments (also: *worlds*, or *states*), that extend infinitely into the future, through which we navigate.
- We may claim that for any moment of time there is exactly one next moment.
- The sequence of moments is called a *(computation/ execution) path*.
- We may also claim that for any moment of time there are several possible next moments.
- This former model is called *linear model of time*, whereas the latter is called *branching model of time*.

Linear vs. branching models of time /cont.





Unary operators in linear-time temporal logic

- Linear-time temporal logic (LTL) introduces the unary temporal operators $\Box$ (*always*), $\Diamond$ (*eventually*), and $\bigcirc$ (*next*).

Formula	**Meaning**
$\Box\phi$	ϕ is true now and in *all* future moments.
$\Diamond\phi$	ϕ is true in *some* moment in time (i.e. now or in the future).
$\bigcirc\phi$	ϕ is true in the *next* moment in time.

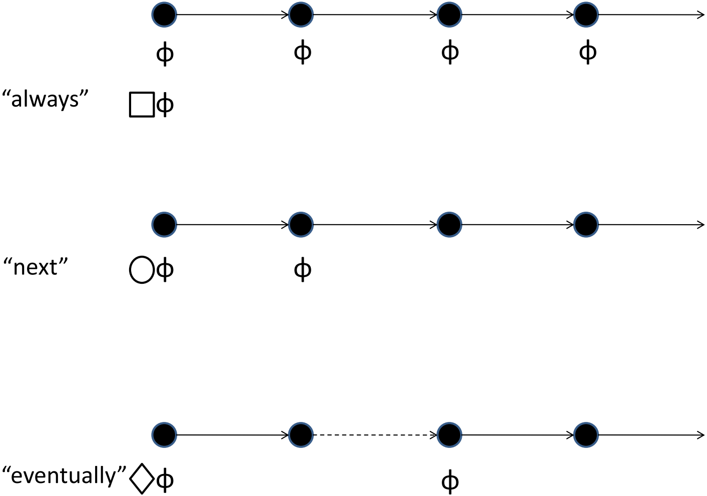
Alternative (textual) notation for unary operators in linear-time temporal logic

Formula	**Alternative notation**
$\Box\phi$	Gϕ (**G**lobal)
$\Diamond\phi$	Fϕ (**F**uture)
$\bigcirc\phi$	Xϕ (Ne**X**t)

The G(lobal) and F(uture) operators

- The G(lobal) operator (denoted by the square) is a universal operator and it translates to “for ANY time t (along the execution path)”
- The G(lobal) operator is expressed by “always.”
- The F(uture) operator (denoted by the diamond) is an existential operator and it translates to “for some time t (along the execution path).”
- The F(uture) operator is expressed by “eventually” or “sometime.”

Temporal unary operators



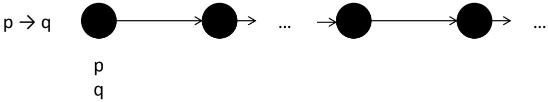
“always” $\Box\phi$

“next” $\bigcirc\phi$

“eventually” $\Diamond\phi$

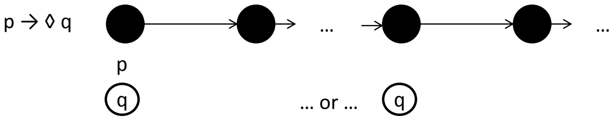
The conditional statement $p \rightarrow q$

- At the time when p is *true*, then q is *true*.



The conditional statement $p \rightarrow \Diamond q$

- Starting from the time when p is *true*, q will be *true* eventually.



Capturing k^{th} future moment in time

Formula	**Meaning**
$\bigcirc \bigcirc \phi$ or $\bigcirc^2$	ϕ is true in two moments of time.
$\bigcirc \bigcirc \bigcirc \phi$ or $\bigcirc^3$	ϕ is true in three moments of time.

Capturing k^{th} future moment in time /cont.

- For example, for the following sequence of statements

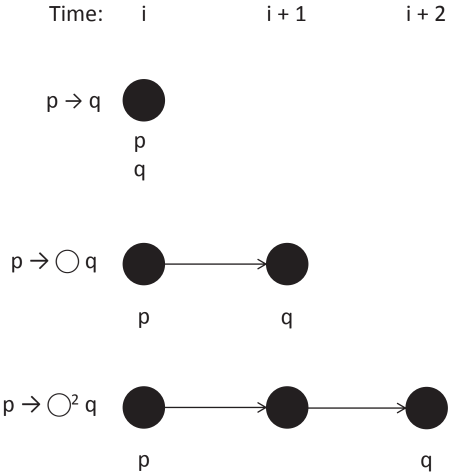
  $$\langle a, b, c \rangle$$

  we have

  $$Start \rightarrow a \wedge \bigcirc b \wedge \bigcirc \bigcirc c$$

Example: The next temporal operator
Expressions and their visualizations

- Consider the 3 formulas below and their corresponding visualizations:



Time: i i + 1 i + 2

$p \rightarrow q$ p q

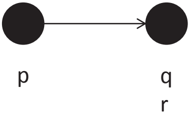
$p \rightarrow \bigcirc q$ p q

$p \rightarrow \bigcirc^2 q$ p q

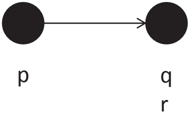
Example: The next temporal operator
Expressions with identical visualizations

- What is the difference between the two expressions?

$(p \land \bigcirc q) \rightarrow \bigcirc r$



$p \rightarrow \bigcirc (q \land r)$

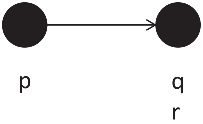


- Note that we evaluate the *entire* expression at some time *i*.
- A formula is true at time *i* if the implication statement is true.

Example: The next temporal operator
Expressions with identical visualizations /cont.

- In $(p \wedge \bigcirc q) \rightarrow \bigcirc r$, the LHS is true at time i when p is true at time i and q is true at time $i + 1$.
- The RHS is true at time i when $\bigcirc r$ is true at time i, i.e. when r is true at time $i + 1$.

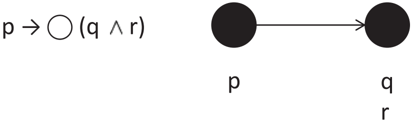
$$(p \wedge \bigcirc q) \rightarrow \bigcirc r$$



- In other words, if p is true at time i and q is true at time $i + 1$, then r is true at time $i + 1$.

The next temporal operator
Expressions with identical visualizations /cont.

- In $p \rightarrow \bigcirc(q \wedge r)$ the LHS is true at time i when p is true at time i, then *both* and q and r are true at time $i + 1$.



Capturing “infinitely often”

- Note that $\Diamond\phi$ implies that ϕ is *eventually* true (i.e. in *some* future moment in time) but there is no claim as to what happens after that, i.e. there is no assumption that ϕ remains true subsequently.
- If we want to state that ϕ becomes true infinitely often (i.e. it becomes true at some future moment, and may subsequently become false, but it may then become true again in the future, etc.) we write $\Box\Diamond\phi$.

Duality of unary operators

- The unary temporal operators $\Box$ and $\Diamond$ are duals since

  $$\neg\Box\phi \equiv \Diamond\neg\phi$$

  In other words, not being the case that ϕ is true in all moments is equivalent to saying that in some moment ϕ will become false.

- Alternatively,

  $$\Diamond\phi \equiv \neg\Box\neg\phi$$

  In other words, saying that in some moment ϕ will be true is equivalent to saying that it is not the case that ϕ will be false in all moments.

Examples with unary operators

- Consider the truthfulness of the following statements:
- $\Box$ Saturday. $\times$
- $\Diamond$ Saturday. ✓
- $\Box(Saturday \rightarrow \bigcirc Sunday)$. ✓

Examples with unary operators /cont.

- We can deploy the unary temporal operators to formalize requirements as in the following examples:
- "The backup server never shuts down."
  $\Box$ (backup server up)
- "A process will eventually execute (thus avoiding starvation)."
  $\Diamond$ (process running)

Examples with unary operators /cont.

- "The sun will rise eventually."
  $\Diamond$ (sun rises)

- "The sun will rise again and again."
  $\Box\Diamond$ (sun rises)

- "Eventually p holds forever."
  $\Diamond\Box p$

Examples with unary operators /cont.

- "Sunrise always leads to sunset."

  $\Box(\text{sun rise} \rightarrow \Diamond\ \text{sunset})$

- "Whenever ϕ holds, also ψ must hold at that moment in time and subsequently."

  $\Diamond(\phi \rightarrow \Box\psi)$

Examples with unary operators /cont.

- "If the level of radiation reaches a certain threshold at state $s_i, i \geq 0$, then at state s_{i+1} an alarm goes off."

  (radiation level reaches threshold) $\rightarrow \bigcirc$ (alarm off)

- "It is always the case that if an sms is sent through the network, it will be eventually delivered or the sender will receive a notification for failure."

  $\Box$(sms sent $\rightarrow \Diamond$ (sms delivered $\oplus$ failure notification sent))

Examples with unary operators /cont.

- “It is always the case that after pressing the eject button, the pilot seat is propelled out of the aircraft.”

  $\Box$ (eject button $\rightarrow \bigcirc$ pilot seat propelled out)

- “It is always the case that once the machine accepts a valid ticket and payment, it will eventually date stamp the ticket.”

  $\Box$(valid ticket $\wedge \bigcirc$ payment $\rightarrow \Diamond$ date stamp ticket)

Examples with unary operators /cont.

- “If a process is put to wait, it will eventually be notified to proceed.”

  (process wait) $\rightarrow$ $\Diamond$ (process woken up)

Examples with unary operators /cont.

- Consider the sentence “It is not always the case that the system remains on.” The sentence translates to

  $\neg \Box$ (system on)

- Let us re-write the formula using the $\Diamond$ operator. The formula is logically equivalent to

  $\Diamond \neg$ (system on)

- Let us now translate our answer from its formal representation back to English. The formula reads “Eventually the system will be off.”

Examples with unary operators /cont.

- "It is always the case that once the juke box receives a valid payment followed immediately by a song selection, it will eventually play the song."

  $\Box\ ((\text{valid payment}) \wedge \bigcirc\ (\text{song selection}) \rightarrow \Diamond\ (\text{play song}))$

- "It is always the case that when smoke is detected, the alarm will go off at the immediately next moment in time."

  $\Box\ ((\text{smoke detected}) \rightarrow \bigcirc\ (\text{alarm off}))$

Binary operators in linear-time temporal logic

- The binary operators Until and Unless are defined as follows:

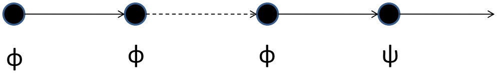
Formula	Meaning
$\phi \mathcal{U} \psi$	ϕ is true up and until some future moment when ψ becomes true. Note that in this case, ψ is guaranteed to become true at some future time.
$\phi \mathcal{W} \psi$	ϕ is true unless ψ becomes true. Note that ψ may never become true. The operator $\mathcal{W}$ is referred to as a “weak until” (as opposed to $\mathcal{U}$ which is referred to as “strong until.”).

Binary operators in linear-time temporal logic /cont.

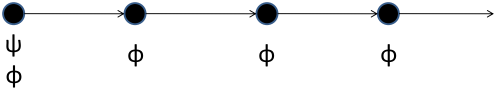
- The binary operators Since and Release are defined as follows:

Formula	**Meaning**
$\phi \mathcal{S} \psi$	ψ has happened sometime in the past and ϕ has been continuously held since then.
$\phi \mathcal{R} \psi$	ψ has to be true at all states until and including the state at which ϕ first becomes true. If ϕ never becomes true, then ψ must remain true forever.

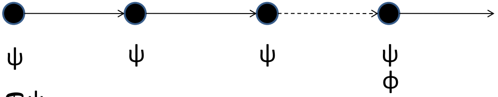
Visualization of binary operators in linear-time temporal logic



“until” $\phi\, \mathcal{U}\, \psi$



“since” $\phi\, \mathcal{S}\, \psi$



“release” $\phi\, \mathcal{R}\, \psi$

Operator precedence

- Unary operators are equally strong and have higher precedence than binary operators.
- For example $\neg\phi\mathcal{U}\psi$ is interpreted as $(\neg\phi)\ \mathcal{U}\ \psi$.
- Furthermore, temporal operators have higher precedence over operators from propositional logic $\wedge$, $\vee$, and $\rightarrow$.
- As in propositional and predicate logic, precedence rules in temporal logic can be overriden by parentheses.

A summary of operator precedence rules

Operator	Precedence
$\neg\ \Box\ \Diamond\ \bigcirc$	1: Highest precedence; Equally strong.
$\mathcal{U}\ \mathcal{W}\ \mathcal{S}\ \mathcal{R}$	2: Equally strong.
$\wedge$	3
$\vee$	4
$\rightarrow$	5
$\leftrightarrow$	6: Lowest precedence.

Operator precedence /cont.

- Consider the following statement:

$$\Diamond p \rightarrow \Box q \vee \neg r \mathcal{R} p$$

- Despite the existence of well-defined precedence rules, statements such as the above can be difficult to read. We can use parentheses to make precedence explicit:

$$\Diamond p \rightarrow [(\Box q) \vee (\neg r \mathcal{R} p)]$$

Well-formed formulas

- In our discussion on propositional logic, we inductively defined a (well formed) formula as follows:
- Each propositional variable on its own is a formula (called *atomic formula*).
- If ϕ is a formula, then $\neg\phi$ is a formula.
- If ϕ and ψ are formulas, and $\bullet$ is any binary connective, then $(\phi \bullet \psi)$ is a formula, where $\bullet$ could be any of (but is not limited to) the usual operators $\wedge, \vee, \rightarrow, \leftrightarrow$.

Well-formed formulas /cont.

- The set of structural rules is referred to as a *grammar* and can help us to determine the correctness of formulas. We can extend the above discussion by adding the vocabulary of temporal logic as follows:
- If ϕ is a formula, then $\mathcal{T}\phi$ where $\mathcal{T}$ is a temporal operator is a formula.
- If ϕ and ψ are formulas, then so are $(\phi \vee \psi)$, $(\phi \wedge \psi)$, and $(\phi \rightarrow \psi)$.

Examples of well-formed formulas

- $\neg\phi$, $\Box\phi$, $\Diamond\phi$ and $\bigcirc\phi$ are (well formed) formulas.
- $\phi \to \Box\Diamond(\psi \vee \chi)$ is a (well formed) formula.
- $\phi \bigcirc \psi$ is not a formula.

Interpretation and properties of formulas

Formula	Interpretation	Property
$\Box p$	Always p.	Invariance.
$\Diamond p$	Eventually p.	Guarantee.
$p \rightarrow (\Diamond q)$	p implies eventually q.	Response.
$p \rightarrow (q \mathcal{U} r)$	p implies q until r.	Precedence
$\Box\Diamond p$	Always eventually p.	Recurrence.
$\Diamond\Box p$	Eventually, always p.	Stability.
$\Diamond p \rightarrow \Diamond q$	Eventually p implies eventually q.	Correlation.

Interpretation of wff's - 18 common patterns of expressions: Example 1

$\phi \rightarrow \psi$

if ϕ is true at time *i*
then ψ is true at time *i*.

Interpretation of wff's: Example 2

$\Box\phi$

ϕ is globally true.

Interpretation of wff's: Example 3

$\Box(\phi \wedge \psi)$

$(\phi \wedge \psi)$ is globally true.

Interpretation of wff's: Example 4

$\Box(\phi \to \psi)$

$(\phi \to \psi)$ is globally true.
It is always the case that **if** ϕ is true **then** ψ is also true at the same time.

Interpretation of wff's: Example 5

$\Box\phi \rightarrow \psi$

if ϕ is globally true
then ψ is true at time *i*.

Interpretation of wff's: Example 6

$\phi \rightarrow \Box\psi$

if ϕ is true at time *i*
then ψ is globally true.

Interpretation of wff's: Example 7

$\phi \to \bigcirc\psi$

if ϕ is true at time *i*
then $\bigcirc\psi$ is true at time *i*.

$\therefore$ ψ is true at time $i + 1$.

Interpretation of wff's: Example 8

$\phi \rightarrow \bigcirc^2 \psi$

if ϕ is true at time *i*
then $\bigcirc^2 \psi$ is true at time *i*.

$\therefore$ ψ is true at time $i + 2$.

Interpretation of wff's: Example 9

$\Box(\phi \rightarrow \bigcirc\psi)$

$(\phi \rightarrow \bigcirc\psi)$ is globally true.

It is always the case that **if** ϕ is true
then $\bigcirc\psi$ is also true at the same time.

$\therefore$ If ϕ is true at some moment, then ψ is also true at the next moment.

Interpretation of wff's: Example 10

$\phi \to \Diamond\psi$

if ϕ is true at time *i*
then $\Diamond\psi$ is true at time *i*.

$\therefore$ ψ becomes true at some moment in time along the path, starting from time *i*.

Interpretation of wff's: Example 11

$\phi \to \bigcirc\Diamond\psi$

if ϕ is true at time i
then $\bigcirc\Diamond\psi$ is true at time i, thus $\Diamond\psi$ is true at time $i+1$.

$\therefore$ ψ is true at some moment in time starting from $\mathit{time} = i + 1$.

Interpretation of wff's: Example 12

$\phi \rightarrow \Diamond\Box\psi$

if ϕ is true at time *i*
then $\Diamond\Box\psi$ is true at time *i*.
$\Box\psi$ is true at some moment in time along the path, starting from *time* = *i*.

$\therefore$ ψ becomes globally true at some moment in time.

Interpretation of wff's: Example 13

$(\phi \land \bigcirc\psi) \rightarrow \bigcirc\tau$

if $\phi \land \bigcirc\psi$ is true at time *i*
then $\bigcirc\tau$ is true at time *i*.

$\therefore$ **if** ϕ is true at time *i* and ψ is true at time $i + 1$
then τ is true at time $i + 1$.

Interpretation of wff's: Example 14

$\phi \rightarrow \bigcirc(\psi \wedge \tau)$

if ϕ is true at time *i*
then $\bigcirc(\psi \wedge \tau)$ is true at time *i*.

$\therefore$ **if** ϕ is true at time *i*
then $(\psi \wedge \tau)$ is true at time $i + 1$.

Interpretation of wff's: Example 15

$\phi \, \mathcal{U} \, \psi$

ϕ is true at time *i* and remains true up until (but *not* including) the moment when ψ first becomes true.

Interpretation of wff's: Example 16

$\phi \rightarrow (\psi \, \mathcal{U} \, \tau)$

if ϕ is true at time *i*
then $(\psi \, \mathcal{U} \, \tau)$ is true at time *i*.

$\therefore$ **if** ϕ is true at time *i*
then ψ becomes true at time *i* and remains true up until (but *not* including) the moment when τ first becomes true.

Interpretation of wff's: Example 17

$\phi \rightarrow \bigcirc(\psi \, \mathcal{U} \, \tau)$

if ϕ is true at time *i*
then $\bigcirc(\psi \, \mathcal{U} \, \tau)$ is true at time *i*, or
$(\psi \, \mathcal{U} \, \tau)$ is true at time $i + 1$.

$\therefore$ **if** ϕ is true at time *i*
then ψ becomes true at time $i + 1$ and remains true up until (but *not* including) the moment when τ first becomes true.

Interpretation of wff's: Example 18

$\phi \mathcal{R} \psi$

ψ is true at time *i* and remains true up and including the moment when ϕ first becomes true.
If ϕ never becomes true, then ψ remains true.

Examples with binary operators /cont.

- "The microwave remains on until a set time has elapsed or it remains on until the button is pressed to open its door."

  (microwave on) $\mathcal{U}$ (set time elapsed $\vee$ button pressed)

- "A process continuously executes unless there is a stack overflow (too much memory is used on the call stack)."

  process executes $\mathcal{W}$ stack overflow.

- "The autopilot system has remained on since it has been activated."

  autopilot on $\mathcal{S}$ autopilot activated.

Examples with binary operators /cont.

- “Upon initiating the landing procedure, the autopilot system is released (or: the autopilot system has to be true at all times until and including the state at which the landing procedure is first initiated).”

  (landing procedure initiated) $\mathcal{R}$ (autopilot on)

Examples with binary operators /cont.

- Consider the sentence “Eventually the process will execute.” The sentence translates to

  $\Diamond$ (process executes)

- Let us re-write the formula using the $\Box$ operator. The formula is logically equivalent to

  $\neg \Box \neg$ (process executes)

- Let us now translate our answer from its formal representation back to English. The formula reads “It is not always the case that the process will not execute.”

Examples with binary operators /cont.

- Consider the sentence "A procedure executes unless it throws an exception." which is translated as

  (procedure executes) $\mathcal{U}$ (procedure throws exception)

- What is wrong with the translation? Let us provide an appropriate formal representation of the sentence.
- The binary operator $\mathcal{U}$ guarantees that the second formula will occur.
- The requirement, however, leaves open the possibility that the formula might not occur.
- The proper representation is

  (procedure executes) $\mathcal{W}$ (procedure throws exception)

Examples with binary operators /cont.

- “Only one server, either the main one or the backup one, will always be up.”

  $\Box$ ((main server up) $\oplus$ (backup server up))

- “In a peer-to-peer network it is always the case that a resource lookup will eventually either discover a viable peer or result in a failure notification.”

  $\Box$ ((resource lookup) $\rightarrow$ $\Diamond$ ((viable peer discovery) $\oplus$ (failure notification)))

Examples with binary operators /cont.

- "It is always the case that the system remains at state *active* until 08:00 or it remains active unless it is disabled."

  $\Box$ ((active $\mathcal{U}$ (time becomes 08:00)) $\vee$ (active $\mathcal{W}$ (system disabled)))

- "The server has been up since it has been switched on."

  (server up) $\mathcal{S}$ (server switched on)

Examples with binary operators /cont.

- "The car remains at cruise control *until and including* the moment when the driver indicates his/her intention to obtain manual control."

  (intention to obtain manual control) $\mathcal{R}$ (cruise control)

Examples with binary operators /cont.

- “The shared resource has remained active since it has been instantiated.”

  (shared resource active) $\mathcal{S}$ (shared resource instantiated)

- “A client process executes continuously unless it is disabled.”

  (client process executes) $\mathcal{W}$ (client process is disabled)

Examples with binary operators /cont.

- "Once its task is completed, a writer process will release the lock to a shared resource at the immediately next moment."

  (writer completes its task) $\rightarrow \bigcirc$ (writer releases lock)

- "A shared resource remains locked by a writer process *until and including* the moment when the writer process completes its task."

  (writer process completes) $\mathcal{R}$ (shared resource locked)

Properties of concurrent and reactive systems: Safety

- **Safety** implies that *something bad will not happen*.
- “The car door must remain closed until the train stops.”
- “A shared buffer must not be simultaneously accessed by a writer and a reader process (or threads of execution).” This particular instance of safety is called *mutual exclusion* (between writers and readers).

Properties of concurrent and reactive systems: Safety /cont.

- “A shared buffer must not be simultaneously accessed by more than one writers.”
  This particular safety property is called *self exclusion* (for the writers).
- “No two processes will be blocked, where each one waiting for the other to terminate before proceeding.”
  This is called *deadlock freedom*.

Modeling safety

- The key temporal concept here is *never* and a key symbol is $\Box\neg$.
- "The reactor temperature must not exceed 1000 degrees."

  $\Box\neg\ (\mathit{reactor\ temperature} > 1000)$

Properties of concurrent and reactive systems: Liveness

- **Liveness** implies that *something good will eventually happen*.
- "A thread (or process) eventually gets access to a shared resource in order to accomplish its task."

Modeling liveness

- The key temporal concept here is *at least once* and a key symbol is $\Diamond$.
- “A process must eventually terminate.”

  $\Diamond\ (\mathit{process\ terminates})$

Properties of concurrent and reactive systems: Fairness

- **Fairness** implies that *a request for a service will eventually be acknowledged and an action is performed*."
- Fairness is typically needed to prove liveness.

Properties of concurrent and reactive systems: Fairness /cont.

- "A writer or reader client of a shared buffer does not wait forever to be serviced."
  This is called *starvation freedom*.
- This is important if we wish to attempt something *continuously*.
- In this case, we would like to know wether it will succeed *never*, *once*, *every time*, or *infinitely often*.
- This distinction is important when describing the scheduling of processes, the response to messages, etc.

Properties of concurrent and reactive systems: Fairness /cont.

- The three types of fairness are:
- *Weak fairness* implies that if a process is continously requested/attempted, then it has to be executed infinitely often.
- *Strong fairness* implies that if a process is requested/attempted infinitely often, then it will execute infinitely often.
- *Unconditional fairness* implies that a process executes infinitely often.

Properties of concurrent and reactive systems: Modeling fairness

1. “If we attempt something infinitely often, then we will succeed infinitely often.”
2. “If we attempt something infinitely often, then we will succeed at least once.”
3. “If we attempt something continuously, then we will succeed infinitely often.”
4. “If we attempt something continuously, then we will succeed at least once.”

Properties of concurrent and reactive systems: Modeling fairness /cont.

- "If we attempt something infinitely often, then we will succeed infinitely often."

  $\Box\Diamond\ \mathit{attempt} \rightarrow \Box\Diamond\ \mathit{success}$

- "If we attempt something infinitely often, then we will succeed at least once."

  $\Box\Diamond\ \mathit{attempt} \rightarrow \Diamond\ \mathit{success}$

Properties of concurrent and reactive systems: Modeling fairness /cont.

- "If we attempt something continuously, then we will succeed infinitely often."

  $\Box\, \mathit{attempt} \rightarrow \Box\Diamond\, \mathit{success}$

- "If we attempt something continuously, then we will succeed at least once."

  $\Box\, \mathit{attempt} \rightarrow \Diamond\, \mathit{success}$

Properties of concurrent and reactive systems: Modeling fairness /cont.

- Statement [4] is weaker than [1]. Why?
  1. $\Box\Diamond\ attempt \rightarrow \Box\Diamond\ success$
  2. $\Box\Diamond\ attempt \rightarrow \Diamond\ success$
  3. $\Box\ attempt \rightarrow \Box\Diamond\ success$
  4. $\Box\ attempt \rightarrow \Diamond\ success$
- In [4] even though attempts are continuously made, we are only guaranteed to succeed at least once.
- In [1] there will be an infinite number of successes, though not necessarily continuous.

Critical sections

- A *critical section* is a code segment that accesses a shared resource (e.g. a data structure) that must not be simultaneously accessed by more than one thread of execution (safety property).
- If one thread is currently in the critical section, another thread that wishes to obtain access must wait.
- As a result, two threads that wish to access a critical section must operate in *mutual exclusion*.

Critical sections /cont.

- A thread is expected to be often in its critical section (liveness property).
- Furthermore, a thread waiting to access a critical section must eventually obtain access (fairness property).

Critical sections /cont.

- Let criti_i denote that thread_i is in the critical section, and wait_i denote that thread_i is waiting.
- The **safety property** can be expressed as

$$\Box(\neg \mathit{criti}_1 \vee \neg \mathit{criti}_2)$$

Critical sections /cont.

- What if we said:

$$\Box(\mathit{criti}_1 \oplus \mathit{criti}_2)$$

Critical sections /cont.

- What if we said:

$$\Box(criti_1 \oplus criti_2)$$

- This would imply that one thread must *always* be in the critical section which is not a requirement.

Critical sections /cont.

- The **liveness property** can be expressed as

$$(\Box\Diamond\ criti_1) \wedge (\Box\Diamond\ criti_2)$$

- The **fairness property** can be expressed as

$$(\Box\Diamond wait_1 \rightarrow \Box\Diamond criti_1) \wedge (\Box\Diamond wait_2 \rightarrow \Box\Diamond criti_2)$$

Example: Concurrent resource

- Consider the case of a shared (concurrent) resource.
- There are two types of clients: Writers access the resource in order to modify its state (i.e. “write”), whereas readers access the resource in order to observe its state (i.e. “read”).
- The system allows multiple readers accessing the resource at the same time.
- However, only one writer may access the resource at a time.
- Additionally, a reader and a writer cannot access the resource at the same time.

Example: Concurrent resource /cont.

- $attempt_{writer}$ and $attempt_{reader}$ respectively denote a writer or a reader attempting to obtain access to the resource.
- $wait_{writer}$ and $wait_{reader}$ respectively denote a writer or a reader waiting, and
- $succeed_{writer}$ and $succeed_{reader}$ respectively denote a writer or a reader succeeding (i.e. being in their critical section).

Example: Concurrent resource /cont.

- "The resource must not be simultaneously accessed by a writer and a reader."
- This is a particular instance of the **safety property** called **mutual exclusion** (between writers and readers).

$$\Box(\neg \mathit{succeed}_{\mathit{writer}} \vee \neg \mathit{succeed}_{\mathit{reader}})$$

Example: Concurrent resource /cont.

- “Each client is infinitely often in its critical section.”
  This refers to **liveness**.

  $(\Box\Diamond \mathit{succeed}_{\mathit{writer}}) \land (\Box\Diamond \mathit{succeed}_{\mathit{reader}})$

Example: Concurrent resource /cont.

- "If a client attempts to obtain access continuously then a client succeeds at least once."
- This refers to **fairness**.

$$(\Box \mathit{attempt}_{\mathit{writer}} \rightarrow \Diamond \mathit{succeed}_{\mathit{writer}}) \wedge (\Box \mathit{attempt}_{\mathit{reader}} \rightarrow \Diamond \mathit{succeed}_{\mathit{reader}})$$

Example: Concurrent resource /cont.

- “A client executes infinitely often.” This is a particular type of **fairness** called **unconditional fairness**.

  $(\Box\Diamond succeed_{writer}) \wedge (\Box\Diamond succeed_{reader})$

Example: Concurrent resource /cont.

- "If a client attempts to obtain access infinitely often then a client succeeds infinitely often."
- This is a particular type of **fairness** called **strong fairness**.

$$(\Box\Diamond \mathit{attempt}_{\mathit{writer}} \rightarrow \Box\Diamond \mathit{succeed}_{\mathit{writer}}) \wedge (\Box\Diamond \mathit{attempt}_{\mathit{reader}} \rightarrow \Box\Diamond \mathit{succeed}_{\mathit{reader}})$$

Example: Concurrent resource /cont.

- "If a client attempts to obtain access continuously then a client succeeds infinitely often."
- This is a particular type of **fairness** called **weak fairness**.

$$(\Box attempt_{writer} \rightarrow \Box\Diamond succeed_{writer}) \wedge (\Box attempt_{reader} \rightarrow \Box\Diamond succeed_{reader})$$

Visualizing patterns of behavior
Example 1

- Consider a program specification given by the following temporal formula:

$$\Box \left[\begin{array}{c} \textbf{start} \rightarrow a \\ \textbf{start} \rightarrow b \\ (a \wedge b) \rightarrow \bigcirc(c \vee d) \\ c \rightarrow \bigcirc c \\ d \rightarrow \bigcirc e \end{array} \right]$$

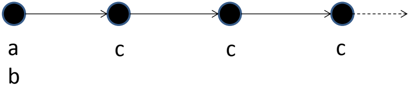
- In the notation we normally omit conjunction among the various expressions of the formula.
- This is interpreted as
  - At the beginning of the program, both a and b will be true.
  - If $(a \wedge b)$ is true, then $(c \vee d)$ is true in the next moment of time.
  - If c is true, then c is true in the next moment of time.
  - If d is true, then e is true in the next moment of time.

Visualizing patterns of behavior Example 1 /cont.

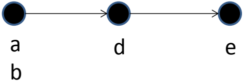
- Given a specification such as this, we want to properly interpret and visualize the possible execution sequences associated with it.
- The key formula here is $(a \wedge b) \rightarrow \bigcirc(c \vee d)$ which reads "When a and b are both true, then in the immediately next moment in time there is a choice of making at least one of c or d true."

Visualizing patterns of behavior
Example 1 /cont.

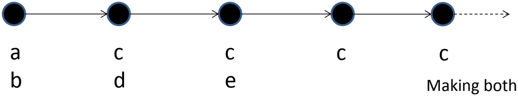
- A model of the specification describes the behavior of the program, since it corresponds to a possible execution.
- In this example, there are three possible models of the specification which we can visualize.



Making c true ensures that c continues forever.



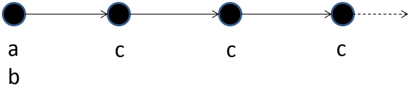
Making d true forces e to be true in the next moment.



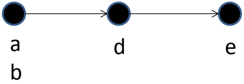
Making both c and d true.

Visualizing patterns of behavior
Example 1 /cont.

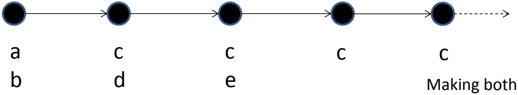
- The formula has two models with infinite behavior.
- The program will terminate upon the occurrence of the sequence $\langle (a \wedge b), d, e \rangle$.



Making c true ensures that c continues forever.



Making d true forces e to be true in the next moment.



Making both c and d true.

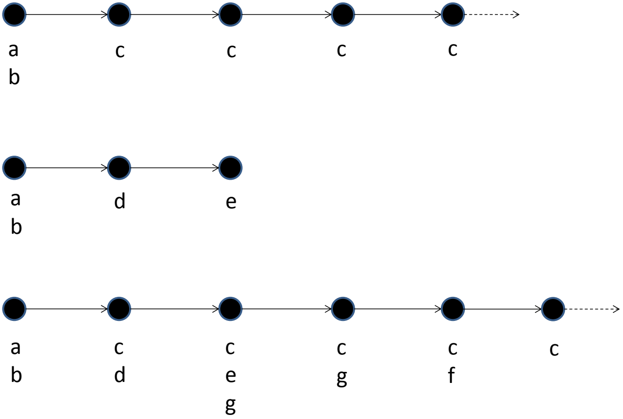
Visualizing patterns of behavior
Example 2

- Consider the following temporal formula:

$$\square \begin{bmatrix} \textbf{start} \rightarrow a \\ \textbf{start} \rightarrow b \\ (a \wedge b) \rightarrow \bigcirc(c \vee d) \\ c \rightarrow \bigcirc c \\ d \rightarrow \bigcirc e \\ (c \wedge e) \rightarrow \bigcirc \bigcirc f \\ (c \wedge d) \rightarrow \bigcirc(g \mathcal{W} f) \end{bmatrix}$$

Visualizing patterns of behavior Example 2 /cont.

- The program will terminate upon the occurrence of the sequence $\langle (a \wedge b), d, e \rangle$.



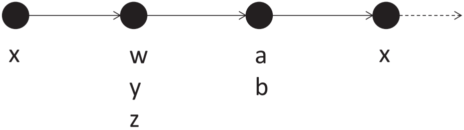
Visualizing patterns of behavior
Example 3

- Given the following temporal formula, what pattern of behavior does it specify?

$$\Box \begin{bmatrix} \textbf{start} \rightarrow x \\ x \rightarrow \bigcirc(w \wedge y \wedge z) \\ z \rightarrow \bigcirc a \\ (w \wedge y \wedge z) \rightarrow \bigcirc b \\ (a \wedge b) \rightarrow \bigcirc x \end{bmatrix}$$

Visualizing patterns of behavior
Example 3 /cont.

- In this example, there is only one model of the specification which we can visualize.



- The program reproduces the pattern $\langle x, (w \wedge y \wedge z), (a \wedge b)\rangle$ indefinitely.