Glossary of notation for "Limits of Computation" module

Symbol	Name	Description
Sets and functions:		
$\{-\}$	set brackets	E.g. $\{a, b, c\}$ is the set containing three elements a, b and c
$\{- \mid -\}$	set comprehension	$\{x \mid P(x)\}$ is the set of objects x having property $P(x)$ E.g. $\{x \mid x \in \mathbb{N} \text{ and } x > 3\} = \{4, 5, 6, \ldots\}$
$\mathbb{N}$	natural numbers	The set of the natural numbers $0, 1, 2, 3, \ldots$
$\mathbb{D}$	set of binary trees	The set of binary trees built from *nil* and $(- \,.\, -)$
$\cup$	union	$A \cup B$ is the set of all elements in either A or B or both E.g. $\{1, 2, 3\} \cup \{2, 5\} = \{1, 2, 3, 5\}$
$\bigcup_{x \in S} A(x)$	indexed union	Means the union of $A(x)$ for all xs in S E.g. $\bigcup_{x \in S} A_x$ means $A_1 \cup A_2 \cup A_3$ $S = \{1, 2, 3\}$
$\subseteq$	subset	$A \subseteq B$ means every element of A is also in B E.g. $\{1\} \subseteq \{1, 2\}$ but not $\{1, 2\} \subseteq \{1\}$
$\subsetneq$	proper subset	$A \subsetneq B$ means that B contains all the elements of A *and some more* elements; A and B cannot be equal
$\times$	Cartesian product	$A \times B$ is the set of pairs of one element from A and one from B E.g. $\{0, 1\} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b)\}$
$\setminus$	set difference	$A \setminus B$ is the set of elements from A which are not in B E.g. $\{0, 1, 2\} \setminus \{1, 3\} = \{0, 2\}$
$\in$	set membership	$x \in A$ means x is in (is an element of) the set A
$\ni$	set membership	$A \ni x$ means x is an element of set A (this is just $x \in A$ written in a different order)
$\to$		$A \to B$ denotes the set of functions from A to B $f : A \to B$ means that f is a function from A to B

Symbol	Name	Description
Sets and functions (continued):		
$\bot$	bottom	$f(x) = \bot$ means that (partial) function f is undefined at x In particular, $\bot$ represents non-termination of programs
$-_\bot$		$X_\bot$ is the set X plus $\bot$ as an additional element Hence $f : A \to B_\bot$ means f is a partial function from A to B
$f(a)\downarrow$		Means that partial function f is defined at a (same as $f(a) \neq \bot$)
$f(a)\uparrow$		Means that partial function f is not defined at a (same as $f(a) = \bot$)
L-programs		The set of programs written in language L
L-data		The set of data values used by programs written in language L
L-store		The set of stores used when giving the semantics of language L
$f^n(x)$		Function f applied n times to x, e.g. $f^3(x)$ is $f(f(f(x)))$
$\lambda x.\ -$	lambda abstraction	$\lambda x.f(x)$ means the function which maps x to $f(x)$ E.g. written this way the squaring function is $\lambda x.\ x \times x$

Symbol	Name	Description
Programming:		
$\Longrightarrow$		Used for rewrite rules (not available in `WHILE`)
$[d_1,\ d_2,\ \cdots,\ d_n]$	list notation	A list of length n, with first element d_1, second element d_2 etc.
$\langle d_1\ .\ d_2 \rangle$		A binary tree with left subtree d_1 and right subtree d_2 In `WHILE`, one uses the **cons** operator: **cons** d_1 d_2
$nil^{\,n}$		A right-spined binary tree containing $n+1$ nils; encodes number n. E.g. $nil^{\,3} = (nil.(nil.(nil.nil)))$

Symbol	Name	Description
Semantics:		
$\{\mathtt{X}_i : \mathtt{d}_i\}$	store	E.g. $\{\mathtt{X} : nil, \mathtt{Y} : (nil.nil)\}$ means a store in which variable `X` contains value nil and variable `Y` contains value $(nil.nil)$
$\sigma[\mathtt{X} := V]$	store update	Means a store that's the same as σ, except that variable `X` has the value V
$[\![-]\!]^L$	semantics of language L	The semantics (meaning) of a programming language L, which is a function $L\text{-}programs \to (L\text{-}data \to L\text{-}data_\perp)$
$[\![p]\!]^L$	semantics of program p	The meaning of a *particular* program p (written in language L) as a function from input to output, i.e. a function $L\text{-}data \to L\text{-}data_\perp$
$\mathscr{E}[\![E]\!]\sigma$		The value of a WHILE expression E in store σ
	For `WHILE` language:	
$C \vdash \sigma \to \sigma'$		Command C terminates when run in store σ, with resulting store σ'
	For machine models:	
$p \vdash s \to s'$		Program (or machine) p transits from state s to state s' in a single step
$p \vdash s \to^* s'$		Program (or machine) p transits from state s to state s' in 0, 1 or more steps
Computation models:		
`TM`		Turing Machines (with tapes and heads etc.)
`GOTO`		Goto language ("flowchart language"); has a "goto" statement instead of the "while" loop statement
`SRAM`		Successor Random Access Machines; has arbitrarily many registers holding natural numbers; allows indirect addressing
`CM`		Counter Machines: much simpler than `SRAM`; machines contain a limited number of counters
`2CM`		Counter Machines with only 2 counters

Symbol	Name	Description
Complexity symbols:		
$time^{\mathtt{L}}_{\mathtt{p}}(\mathtt{d})$	running time	The time taken for a program $\mathtt{p} \in L$-*programs* to run on input $\mathtt{d}$
$\mathcal{T}$		$\mathcal{T}\mathtt{E}$ is the time taken to evaluate `WHILE` expression $\mathtt{E}$
$C \vdash^{time} \sigma \Rightarrow t$		`WHILE` command C terminates after t time steps, when run in store σ'
$L \preceq^{ptime} M$		Language M can simulate language L up to a polynomial difference in time
$L \preceq^{lintime} M$		Language M can simulate language L up to a linear difference in time
$L \preceq^{lintime\text{-}pg\text{-}ind} M$		M can simulate L up to a *program-independent* linear difference in time
$L \equiv^{ptime} M$		Languages L and M are *polynomially equivalent*
$L \equiv^{lintime} M$		Languages L and M are *linearly equivalent*
$L \equiv^{lintime\text{-}pg\text{-}ind} M$		Languages L and M are *strongly linearly equivalent*
$\mathtt{L}^{time(f(n))}$		The set of L-*programs* with time bound f (where $f : \mathbb{N} \to \mathbb{N}$)
$\mathtt{L}^{ptime}$		The set of L-*programs* bounded by some polynomial function
$\mathtt{L}^{lintime}$		The set of L-*programs* bounded by some linear function
$\mathbf{TIME}^{\mathtt{L}}(f)$		The set of decision problems (not programs!) that the language `L` can decide in time f
$\mathbf{P}^{\mathtt{L}}$		The set of decision problems (not programs) that the language `L` can decide in polynomial time
$\mathbf{EXP}^{\mathtt{L}}$		The set of decision problems (not programs) that the language `L` can decide in exponential time
$\mathbf{LIN}^{\mathtt{L}}$		The set of decision problems (not programs) that the language `L` can decide in linear time
$\mathbf{NP}^{\mathtt{L}}$		The set of decision problems accepted by a nondeterministic `L`-program in polynomial time

Symbol	Name	Description
Other symbols:		
$\forall$		"for all"
iff		"if and only if"
$\simeq$		$x \simeq y$ means either x and y are both undefined, or both are defined and they are equal
$\sqsubseteq$	language matching	$S \sqsubseteq T$ means: any program in language S is also a program in language T, and has the same semantics
$\|x\|$	size	Denotes the size of an object (e.g. binary tree) x
$\ulcorner X \urcorner$		The encoding or translation of some object X (see e.g. rules for encoding numbers or programs as binary trees)
$n \dot{-} m$	cutoff subtraction	Gives $n - m$ if $n > m$ and 0 otherwise
X^*	Kleene star	Strings built from 0 or more repetitions of the string expression X as used in regular expressions; e.g. $\{0,1\}^*$ denotes binary strings
ϵ	empty string	ϵ is the empty string, i.e. the string of length 0