CLTL experiments			
Result	**File name**	**Name in the paper**	**Informal description**
S	sorting1.lisp	S1-1	Sorts an array of **integer** numbers of size 5, by iteratively choosing an index i such that a[i] > a[i+1] and swapping the i-th and (i+1)-th elements.
U	sorting2.lisp	S1-2	Property: Eventually the array gets sorted.
S	sorting3.lisp	S1-3	Property: Within 8 time instants the array gets sorted.
S	sorting-r1.lisp	S1-R-1	Sorts an array of **real** numbers of size 5, by iteratively choosing an index i such that a[i] > a[i+1] and swapping the i-th and (i+1)-th elements.
U	sorting-r2.lisp	S1-R-2	Property: Eventually the array gets sorted.
S	sorting-r3.lisp	S1-R-3	Property: Within 8 time instants the array gets sorted.
S	sorting_Sij1-N-k25.lisp	S2-N-1	Sorts an array of **integers** of size N by iteratively choosing two indexes, i and j, such that i < j and a[i] > a[j], and swapping them, until no two such indexes exist in the array.
U	sorting_Sij2-N-k25.lisp	S2-N-2	Property: Eventually the array gets sorted.
S	sorting_Sij3-N-k25.lisp	S2-N-3	Property: Within 8 time instants the array gets sorted.
S	sorting_Sij4-N-k25.lisp	S2-N-4	Property: Within 9 time instants the array gets sorted.
S	sorting_Sij5-N-k25.lisp	S2-N-5	Sorts and asserts that the array gets sorted eventually.
U	sorting_Sij6-N-k25.lisp	S2-N-6	Property: Regardless of the order of elements, resulting array is equal to the given array.
S	leader-Sat-N	L-N-Sat	Leader election without any property. N is the number of nodes.
U	leader-p1-N	L-N-p1	Property: Eventually we have a winner.
U	leader-p2-N	L-N-p2	Property: At most we have one winner.
S	uml-leader-Sat-N	UL-N-Sat	UML leader election without any property. N is the number of nodes.
U	uml-leader-p1-N	UL-N-p1	Property: Eventually we have a winner.
U	uml-leader-p2-N	UL-N-p2	Property: At most we have one winner.
S	CCAS-Sat-1	CC-1-Sat	Warning delay=30 Notification Delay=10 Without any property.
S	CCAS-p1-1	CC-1-p1	Warning delay=30 Notification Delay=10 Property: If the distance is less than 2 for 45 time units, then the system braked sometime in the last 45 time units.
U	CCAS-p2-1	CC-1-p2	Warning delay=30 Notification Delay=10 Property: If the distance is less than 2 for 54 time units, then the system braked sometime in the last 54 time units.
S	CCAS-Sat-2	CC-2-Sat	Warning delay=40 Notification Delay=10 Without any property.

S	CCAS-p1-2	CC-2-p1	Warning delay=40 Notification Delay=10 Property: If the distance is less than 2 for 55 time units, then the system braked sometime in the last 55 time units.
U	CCAS-p2-2	CC-2-p2	Warning delay=40 Notification Delay=10 Property: If the distance is less than 2 for 64 time units, then the system braked sometime in the last 64 time units.
S	CCAS-Sat-3	CC-3-Sat	Warning delay=50 Notification Delay=10 Without any property.
S	CCAS-p1-3	CC-3-p1	Warning delay=50 Notification Delay=10 Property: If the distance is less than 2 for 65 time units, then the system braked sometime in the last 65 time units.
U	CCAS-p2-3	CC-3-p2	Warning delay=50 Notification Delay=10 Property: If the distance is less than 2 for 74 time units, then the system braked sometime in the last 74 time units.
S	CCAS-Sat-4	CC-4-Sat	Warning delay=30 Notification Delay=15 Without any property.
S	CCAS-p1-4	CC-4-p1	Warning delay=30 Notification Delay=15 Property: If the distance is less than 2 for 55 time units, then the system braked sometime in the last 55 time units.
U	CCAS-p2-4	CC-4-p2	Warning delay=30 Notification Delay=15 Property: If the distance is less than 2 for 64 time units, then the system braked sometime in the last 64 time units.
S	CCAS-Sat-5	CC-5-Sat	Warning delay=40 Notification Delay=15 Without any property.
S	CCAS-p1-5	CC-5-p1	Warning delay=40 Notification Delay=15 Property: If the distance is less than 2 for 55 time units, then the system braked sometime in the last 55 time units.
U	CCAS-p2-5	CC-5-p2	Warning delay=40 Notification Delay=15 Property: If the distance is less than 2 for 64 time units, then the system braked sometime in the last 64 time units.

CLTLoc experiments

Result	File name	Name in the paper	Informal description	Formalization
S	Lampada1.cltl	LC1	p3lt is the property.	Property: (defconstant p3lt (->) (somf (&& (-P- L) ([>=] (-V- caux) D))) (somf (&& (-P- On) (|| (&& ([=] (-V- c1) 0) (next (until ([>] (-V- c1) 0) (&& (-P- On) ([<] (-V- c1) D))))) (&& ([=] (-V- c2) 0) (next (until ([>] (-V- c2) 0) (&& (-P- On) ([<] (-V- c2) D))))))))))
U	Lampada2.cltl	LC2	Same as LC1 conjuncted with the "constraints".	Lampada1.cltl && (defconstant constraints (&& (alwf (&& (!! (-P- Off)) (!! (-P- TurnOff)) (-> (-P- On) (alwf_e (!! (-P- On)))))) (somf (-P- On))))
S	Lampada3.cltl	LC3	Property: If there was Light in the previous time instant, the value of the counter is less than or equal to Delta now.	;; Y(L) -> cp <= D (defconstant p1 (->) (yesterday (-P- L)) ([<=] (-V- caux) D)))

U	Lampada4.cltl	LC4	p3 and p3lt are the properties.	Property: ;; :def p3 (-> (F_i+ 0 (&& l (G_ei 0 5 l))) (F_i+ 0 (|| on (F_ei 0 5 on)))) (defconstant p3 (-> (somf (&& (-P- L) ([>=] (-V- caux) D))) (somf (&& (-P- On) (|| (&& ([=] (-V- c1) 0) (next (until ([>] (-V- c1) 0) (&& (-P- On) ([<=] (-V- c1) D))))) (&& ([=] (-V- c2) 0) (next (until ([>] (-V- c2) 0) (&& (-P- On) ([<=] (-V- c2) D))))))))))
U	Lampada5.cltl	LC5	p3 is the property.	
S	Lampadaqtl.cltl	LM1	Without any property.	
S	Lampadaqtlp1.cltl	LM2	K=20 Property: There will be some light in the future (origin is excluded) even if ON is always untouched.	:def p1 (&& (F_e+ 0 l) (G_i+ 0 (!! on)))
S	Lampadaqtlp2.cltl	LM3	K=20 Property: Always neither we have light at the current time instant nor some times in the future time window of (0,5].	:def p2 (G_i+ 0 (|| (!! l) (F_ei 0 5 (!! l))))
U	Lampadaqtlp3.cltl	LM4	K=20 Property: Having light for a time window of the size 5 any time in the execution implies, at least there is a time window of the size 5 in which ON is pressed more than one time.	:def p3 (-> (F_i+ 0 (G_ii 0 5 l)) (F_i+ 0 (&& on (F_ei 0 5 on))))
S	Lampadaqtlp1K200.cltl	LM5	Same as Lampadaqtlp1.cltl with 200 as the K.	
S	Lampadaqtlp2K200.cltl	LM6	Same as Lampadaqtlp2.cltl with 200 as the K.	
U	Lampadaqtlp3K25.cltl	LM7	Same as Lampadaqtlp3.cltl with 25 as the K.	
S	Spikes.cltl	SP1	P is a spike at all time instants.	
S	SpikesNotprop.cltl	SP2	P is a spike at all time instants. Property: P is a right-close signal signal.	:def prop1 (F_e+ 0 (|| (U p p) (U p (!! p))))
S	SpikesPast.cltl	SP3	P is a spike at every 80 time instant. Q follows and precedes P in a time window of size 1. Q is a spike.	
U	SpikesPastAper.cltl	SP4	P is a spike at every 100 time instant. Q follows or precedes P in a time window of size 1. Q implies there is no occurrences of Q within 100 time instants in the future.	
U	SpikesPastAperp11.cltl	SP5	P is a spike at every 100 time instant. Q follows or precedes P in a time window of size 1. Q implies there is no occurrences of Q within 100 time instants in the future. Property: Every occurrence of P is followed by at least an occurrence of Q within 2 time instants in the future.	Property: :def prop11 (G_i+ 0 (-> p (F_ee 0 2 q)))

U	SpikesPastAperp22.cltl	SP6	P is a spike at every 100 time instant. Q follows or precedes P in a time window of size 1. Q implies there is no occurrences of Q within 100 time instants in the future. Property: Every occurrence of Q is followed by at least an occurrence of Q within 2 time instants in the future.	Property: :def prop22 (G_i+ 0 (-> q (F_ee 0 2 q)))
U	SpikesPastp1.cltl	SP7	P is a spike at every 80 time instant. Q follows and precedes P in a time window of size 1. Q is a spike. Property: Every occurrence of P is followed by at least an occurrence of Q within 80 time instants in the future.	Property: :def prop1 (G_i+ 0 (-> p (F_ee 0 80 q)))
U	SpikesPastp2.cltl	SP8	P is a spike at every 80 time instant. Q follows and precedes P in a time window of size 1. Q is a spike. Property: Every occurrence of Q is followed by at least an occurrence of Q within 80 time instants in the future.	Property: :def prop2 (G_i+ 0 (-> q (F_ee 0 80 q)))
S	SpikesPastp3.cltl	SP9	P is a spike at every 80 time instant. Q follows and precedes P in a time window of size 1. Q is a spike. Property: Every occurrence of Q is followed or precedes by another occurrence of Q in the time window of size 2.	Property: :def prop3 (G_i+ 0 (-> q (|| (F_ee 0 2 q) (P_ee 0 2 q))))
S	SpikesPastp4.cltl	SP10	P is a spike at every 80 time instant. Q follows and precedes P in a time window of size 1. Q is a spike. Property: Every occurrence of Q is followed by at least an occurrence of Q within 80 time instants in the past.	Property: :def prop2 (G_i+ 0 (-> q (P_ee 0 80 q)))
S	Spikesprop.cltl	SP11	P is a spike at all time instants and at least one time P maintain its true value continuously or P goes from true to false.	
S	mitl-wave-1.cltl	W1	K=20 P is a square wave.	
S	mitl-wave-2.cltl	W2	K=20 P is a square wave and in the absolute time window of (0,3] having some time window of (0,1] with always P implies that in there is some negated P since then until 2 time instants.	(F_ei 0 3 (-> (G_ei 0 1 p) (F_ei 0 2 (!! p))))
S	mitl-wave-3.cltl	W3	K=20 P is a square wave and some time in the absolute time window of (0,2] there is negated P.	:def ax5 (F_ei 0 2 (!! p))
S	mitl-wave-4.cltl	W4	K=20 P is a square wave and absence of P implies that a time window of length 1 with always P starts within one time instant since then.	:def ax6 (G_e+ 0 (-> (!! p) (F_ei 0 1 (G_ei 0 1 p))))
S	mitl-wave-5.cltl	W5	K=20 P is a square wave and there is absolute time window of (2,3] in which always P holds.	:def ax7 (G_ei 2 3 p)
S	mitl-wave-1K30.cltl	W6	Same as mitl-wave-*.cltl with 30 as the K.	
S	mitl-wave-2K30.cltl	W7		
S	mitl-wave-3K30.cltl	W8		
S	mitl-wave-4K30.cltl	W9		
S	mitl-wave-5K30.cltl	W10		
S	Counting10x-1.cltl	C1-1	Q is a spike. P is a spike every at 10 time instants. Every occurrence of P implies exactly one occurrence of Q within time window of (0,10).	
U	Counting10x-2.cltl	C1-2	Q is a spike. P is a spike every at 10 time instants. Every occurrence of P implies exactly one occurrence of Q within time window of (0,10). Property: Infinitely often P occurs such that it is followed or precedes with an occurrence of Q with a time window of [1,5].	Property: :def p1 (G_e+ 0 (F_i+ 0 (&& p (|| (F_ii 0 5 q) (P_ii 0 5 q)))))

U	Counting10x-3.cltl	C1-3	Q is a spike. P is a spike every at 10 time instants. Every occurrence of P implies exactly two occurrences of Q within time window of (0,10). Property: Infinitely often P occurs such that it is followed or precedes with an occurrence of Q with a time window of [0,5].	Property: :def p1 (G_e+ 0 (F_i+ 0 (&& p (|| (F_ii 0 5 q) (P_ii 0 5 q)))))
S	Counting10x-4.cltl	C1-4	Q is a spike. P is a spike every at 10 time instants. Every occurrence of P implies exactly one occurrences of Q within time window of (0,10). Property: Infinitely often P occurs such that it is followed or precedes with an occurrence of Q with a time window of (1,5).	
U	Counting10x-5.cltl	C1-5	Q is a spikes. P is a spike every at 10 time instants. Every occurrence of P implies exactly two occurrences of Q within time window of (0,10).	
S	CountingJustQ1-k25.cltl	C2-1	Q occurs infinitely often, and every occurrence is followed by at least 2 occurrences of Q within 2 time instants. Property: In any time window of length 1 (both ends are excluded) there is at least an occurrence of Q.	:qtl-i :bound 25 :def live_q (G_i+ 0 (F_e+ 0 q)) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :def prop1 (G_e+ 0 (F_ee 0 1 q)) :formula (&& live_q ax_q (!! prop1))
U	CountingJustQ2-k25.cltl	C2-2	Q occurs infinitely often, and every occurrence is followed by at least 2 occurrences of Q within 2 time instants. Property: Infinitely often Q occurres and there is at least one occurrence of Q since then until next time instant (both ends are excluded).	:qtl-i :bound 25 :def live_q (G_i+ 0 (F_e+ 0 q)) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :def prop2 (G_i+ 0 (F_e+ 0 (&& q (F_ee 0 1 q)))) :formula (&& live_q ax_q (!! prop2))
S	CountingJustQ3-k25.cltl	C2-3	Q occurs infinitely often, and every occurrence is followed by at least 2 occurrences of Q within 2 time instants. Property: Infinitely often if Q occurres there is at least one occurrence of Q since then until next time instant (both ends are excluded).	:qtl-i :bound 25 :def live_q (G_i+ 0 (F_e+ 0 q)) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :def prop3 (G_i+ 0 (-> q (F_ee 0 1 q))) :formula (&& live_q ax_q (!! prop3))
S	CountingJustQ4-k25.cltl	C2-4	Every occurrence is followed by at least 2 occurrences of Q within 2 time instants, there is at least one occurrence of Q. Property: Q occurs infinitely often.	:qtl-i :bound 25 :def live_q (G_i+ 0 (F_e+ 0 q)) :def som_q (F_e+ 0 q) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :formula (&& ax_q som_q (!! live_q))

U	CountingJustQ5-k25.cltl	C2-5	Q is a spike and every occurrence is followed by at least 2 occurrences of Q within 2 time instants. Property: Infinitely often Q occurres and there is at least one occurrence of Q since then until next time instant (both ends are excluded).	:qtl-i :bound 25 :def som_q (F_e+ 0 q) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :def spike_q (G_e+ 0 (-> q (U (!! q) true))) :def prop2 (G_i+ 0 (F_e+ 0 (&& q (F_ee 0 1 q)))) :formula (&& ax_q som_q spike_q (!! prop2))
S	CountingJustQ6-k25.cltl	C2-6	Q occurs infinitely often, and every occurrence is followed by at least 2 occurrences of Q within 2 time instants.	:qtl-i :bound 25 :def live_q (G_i+ 0 (F_e+ 0 q)) :def ax_q (G_i+ 0 (-> q (C 2 2 q))) :formula (&& live_q ax_q)
S	CountingJustQ1-k30.cltl	C2-7	Same as CountingJustQ*-k25 with 30 as the K.	
TO	CountingJustQ2-k30.cltl	C2-8		
S	CountingJustQ3-k30.cltl	C2-9		
S	CountingJustQ4-k30.cltl	C2-10		
TO	CountingJustQ5-k30.cltl	C2-11		
S	CountingJustQ6-k30.cltl	C2-12		
U	Counting-k15	C3-1	P and Q are spikes, P occurs at the origin time (absolute 0) instant and P never occurs in the absolute time window of (0,1), at any time absence of P at (0,1) time window implies absence of P at (1,2) time window. P is followed by another P within 2 time instants (both ends are excluded). Occurrence of P forces absence of Q and have exactly 2 occurrences of Q within 1 time instant.	:qtl-i :bound 15 :def ax1_p (G_i+ 0 (-> (G_ee 0 1 (!! p)) (G_ee 1 2 (!! p)))) :def ax2_p (G_i+ 0 (-> p (F_ee 0 2 p))) :def init_p (&& (G_ee 0 1 (!! p)) p) :def spike_q (G_e+ 0 (-> q (U (!! q) true))) :def ax_pq (G_i+ 0 (-> p (&& (!!q) (C 2 1 q) (!! (C 3 1 q)))) :formula (&& p ax1_p ax2_p init_p spike_q ax_pq)
U	Counting-k20	C3-2	Same as Counting-k15 wih 20 as the K.	
U	FiniteVar.cltl	F1	P is a signal that is not finetly variable (i.e, it can change an infinite number of times in a bounded interval). The specification is unSatisfiabile, since the QTL-to-CLTLoc encoding assumes that signals are finitely variable.	:qtl :bound 10 :def ax1 (G_e+ 0 (|| (U p p) (U (!! p) (!! p)))) :def ax2 (G_e+ 0 (|| (S p p) (S (!! p) (!! p)))) :formula (|| (!! ax1) (!! ax2))
U	FiniteVar-k40.cltl	F2	Same as FiniteVar.cltl with 40 as the K.	
U	FiniteVar-k60.cltl	F3	Same as FiniteVar.cltl with 60 as the K.	

U	squaremitl(k10).cltl	Sq1	P is a square wave such that it is true/false in intervals that are open on the left and closed on the right. The MITL-to-CLTLoc translation used is the one that works under the assumption that intervals are closed to the left and open to the right (i.e., "left closed, right open", or "l.c.r.o"), so the verification returns unSat because of this conflict	:mitl-i :bound 10 :def ax1 (G_i+ 0 (-> (G_ei 0 1 p) (G_ei 1 2 (!! p)))) :def ax2 (G_e+ 0 (-> (G_ei 0 1 (!! p)) (G_ei 1 2 p))) :def ax3 (G_ei 0 1 p) :formula (&& ax3 (&& ax2 ax1))
U	squaremitl-k20.cltl	Sq2	Same as squaremitl with 20 as the K.	