event-structure-theory^{0,22}

ABS: event_system_typename() **event_system_typename**

STM: event_system_typename_wf

ABS: EventsWithOrder **EOrder**

STM: EOrder_wf

ABS: EventsWithState **EState**

STM: EState_wf

STM: EState-subtype-EOrder

ABS: EventsWithKinds **EKind**

STM: EKind_wf

ABS: EventsWithValues **EVal**

STM: EVal_wf

ABS: ESAtomAxiom{i:l}(T;V) **ESAtomAxiom**

ABS: ESMachineAxiom(E;T;V;M;*loc*;*knd*;*val*;*when*;*after*;*sndr*;*Trans*;*Send*;*Choose*) **ESMachineAxiom**

STM: ESMachineAxiom_wf

STM: ESAtomAxiom_wf

STM: ESAtomAxiomTrivial

ABS: ES0 **event-system0**

STM: event-system0_wf

ABS: old_event_system{i:l}() **old_event_system**

ABS: $e = e'$ **es-eq-E**

STM: es-eq-E_wf

STM: assert-es-eq-E

STM: decidable__es-E_equal

ABS: es-LnkTag-deq **es-LnkTag-deq**

STM: es-LnkTag-deq_wf

ABS: case(kind(e))act(a) => f(a)rcv(l,tg) => g(l;tg) **es-kindcase**

STM: es-kindcase_wf

ABS: msgtype(m) **es-msgtype**

STM: es-msgtype_wf

STM: es-valtype-kindtype

ABS: state@i\\x **es-state-without**

STM: es-state-without_wf

STM: es-state-eta

STM: event_system-level-subtype

ABS: mk-es0(E;eq;T;V;M;loc;k;v;w;a;$snds$;$sndr$;i;f;prd;cl;p;q) **mk-es0**

STM: mk-es0_wf

ABS: mk-eval(E;eq;prd;$info$;oax;T;w;a;sax;V;v) **mk-eval**

STM: mk-eval_wf

ABS: AtomFreeDecls(X) **EVal-atom-free**

STM: EVal-atom-free_wf

STM: EVal-to-ES

ABS: EVal_to_ES{i:l}(e,p) **EVal_to_ES**

STM: EVal_to_ES_wf

ABS: state when e\\x **es-state-when-without**

STM: es-state-when-without_wf

ABS: state after e\\x **es-state-after-without**

STM: es-state-after-without_wf

ABS: e c$\leq$ e' **es-causle**

STM: es-causle_wf

STM: es-locl-trans

STM: es-le-trans

STM: es-le-trans2

STM: es-le-trans3

STM: es-index-zero

STM: es-causle-trans

STM: es-causle_transitivity

STM: es-le-causle

STM: es-le-total

STM: es-locl-swellfnd

STM: es-causl-wellfnd

STM: es-le-not-locl

STM: es-causal-antireflexive

STM: es-causl-locl

STM: es-causle-le

STM: es-pred-locl

STM: es-le-self

STM: es-pred-le

STM: es-pred-causl

STM: es-sender-causl

STM: es-sender-causle

STM: es-first-implies

STM: es-loc-rcv

STM: es-isrcv-loc

STM: es-hasloc

STM: es-loc-sender

STM: same-sender-index

STM: es-le-iff

STM: es-first-le

STM: es-le-antisymmetric

STM: es-first-unique

STM: es-causl-iff

STM: implies-es-pred

STM: es-le-pred

STM: implies-es-pred2

STM: es-pred-one-one

STM: decidable__es-locl

STM: es-next

ABS: $e <\text{loc } e'$ **es-bless**

STM: es-bless_wf

STM: assert-es-bless

STM: decidable__es-le

ABS: es-ble{i:l}$(es;e;e')$ **es-ble**

STM: es-ble_wf

STM: assert-es-ble

STM: decidable__es-causl

ABS: es-bc{i:l}$(es;e;e')$ **es-bc**

STM: es-bc_wf

STM: assert-es-bc

ABS: $\exists e{=}k(v).P(e;v)$ **es-ek**

ABS: $\exists e{:}\text{rvc}(l,tg,v).P(e;v)$ **es-er**

ABS: mval(m) **es-mval**

STM: es-mval_wf

STM: es-mval-valtype

STM: es-msg-rcvd

ABS: before(e) **es-before**

STM: es-before_wf

STM: es-before_wf2

STM: member-es-before

STM: l_before-es-before

STM: l_before-es-before-iff

ABS: es-init(es;e) **es-init**

STM: es-init_wf

STM: es-init-le

STM: es-first-init

STM: es-loc-init

ABS: [e, e'] **es-interval**

STM: es-interval_wf

STM: member-es-interval

STM: l_before-es-interval

STM: hd-es-interval

STM: es-interval-non-zero

STM: es-interval-nil

STM: es-interval-is-nil

STM: es-interval-last

STM: es-interval-less

STM: es-interval-less-sq

STM: es-interval-eq

STM: es-interval-eq2

STM: es-interval-length-one-one

STM: es-interval-one-one

STM: es-interval-iseg

STM: es-interval-partition

STM: es-interval-select

STM: es-interval_wf2

ABS: haslnk(l;e) **es-haslnk**

STM: es-haslnk_wf

STM: assert-es-haslnk

ABS: rcvs(l;before(e')) **es-rcvs**

STM: es-rcvs_wf

STM: member-es-rcvs

ABS: snds(l;before(e)) **es-snds**

STM: es-snds_wf

STM: member-es-snds

ABS: snds(l, before(e,n)) **es-snds-index**

STM: es-snds-index_wf

STM: member-es-snds-index

STM: firstn-before

STM: es-before-decomp

STM: last-es-snds-index

ABS: emsg(e) **es-msg**

STM: es-msg_wf

STM: es-msg_wf2

STM: es-msg-member-sends

ABS: msgs(l;before(e')) **es-msgs**

STM: es-msgs_wf

STM: haslink_wf2

STM: member-es-msgs

STM: es-fifo-nil

STM: es-fifo

STM: es-after-pred

STM: decl-state-exists

STM: decl-state-subtype

ABS: $@i$ always.$P(x)$ **alle-at1**

STM: alle-at1_wf

ABS: $@i$ always.$P(x_1;x_2)$ **alle-at2**

STM: alle-at2_wf

STM: alle-at-iff

STM: alle-at-not-first

STM: es-invariant1

STM: es-invariant2

STM: es-constant1

ABS: $\exists e@i.P(e)$ **existse-at**

STM: existse-at_wf

STM: change-lemma

STM: es-first-exists

STM: change-since-init

ABS: $\exists e{\leq}e'.P(e)$ **existse-le**

STM: existse-le_wf

ABS: $\exists e{<}e'.P(e)$ **existse-before**

STM: existse-before_wf

STM: existse-before-iff

STM: decidable__existse-before

STM: existse-le-iff

STM: decidable__existse-le

ABS: $\forall e'{\geq}e.P(e')$ **alle-ge**

STM: alle-ge_wf

ABS: $\forall e{<}e'.\ P(e)$ **alle-lt**

STM: alle-lt_wf

STM: alle-lt-iff

STM: decidable__alle-lt

ABS: $\forall e{\leq}e'.P(e)$ **alle-le**

STM: alle-le_wf

STM: alle-le-iff

STM: decidable__alle-le

ABS: $\forall e{\in}[e_1,e_2).P(e)$ **alle-between1**

STM: alle-between1_wf

STM: alle-between1-true

STM: alle-between1-false

STM: alle-between1_functionality_wrt_iff

STM: decidable__alle-between1

ABS: $\exists e{\in}[e_1,e_2).P(e)$ **existse-between1**

STM: existse-between1_wf

STM: existse-between1-true

STM: existse-between1-false

STM: existse-between1_functionality_wrt_iff

STM: decidable__existse-between1

ABS: $\forall e{\in}[e_1,e_2].P(e)$ **alle-between2**

STM: alle-between2_wf

STM: alle-between2-true

STM: alle-between2-false

STM: alle-between2_functionality_wrt_iff

STM: decidable__alle-between2

ABS: $\exists e{\in}[e_1,e_2].P(e)$ **existse-between2**

STM: existse-between2_wf

STM: existse-between2-false

STM: existse-between2-true

STM: existse-between2_functionality_wrt_iff

STM: decidable__existse-between2

ABS: $\exists e \in (e_1, e_2].P(e)$ **existse-between3**

STM: existse-between3_wf

STM: existse-between3-false

STM: existse-between3-true

STM: existse-between3_functionality_wrt_iff

STM: decidable__existse-between3

STM: es-subinterval

STM: last-change

ABS: e is first@ i s.t. $e.P(e)$ **es-first-at**

STM: es-first-at_wf

STM: es-first-before

STM: es-first-before2

ABS: es-first-at-since(es;i;e;$e.R(e)$;$e.P(e)$) **es-first-at-since**

STM: es-first-at-since_wf

STM: previous-event-exists

STM: es-first-at-since-iff

ABS: es-first-at-since'(es;i;e;$e.R(e)$;$e.P(e)$) **es-first-at-since'**

STM: es-first-at-since'_wf

ABS: $\forall e{=}\text{rcv}(l,tg).\ P(e)$ **alle-rcv**

STM: alle-rcv_wf

ABS: $\exists e{=}\text{rcv}(l,tg).\ P(e)$ **existse-rcv**

STM: existse-rcv_wf

STM: es-bound-list

STM: es-bound-list2

STM: es-machine-axiom

STM: atom-free-es-kindtype

STM: atom-free-es-Msg

STM: atom-free-es-valtype

STM: atom-free-es-vartype

STM: atom-free-es-state

STM: atom-free-es-state-without

ABS: e receives a **es-rcv-atom**

STM: es-rcv-atom_wf

ABS: e sends a **es-send-atom**

STM: es-send-atom_wf

ABS: e sends a to i **es-send-atom-to**

STM: es-send-atom-to_wf

ABS: e leaks x to e' **es-leaks**

ABS: e copies x **es-copies**

STM: state-after-pred

STM: implies-es-atom-axiom

ABS: $i >> a$ **es-atom**

STM: es-atom_wf

STM: es-copies_wf

STM: es-leaks_wf

STM: es-atom-axiom

STM: es-atom-lemma1

STM: es-atom-lemma2

ABS: $@i$ stable $state.P(state)$ **es-stable**

STM: es-stable_wf

STM: stable-implies

STM: stable-implies2

STM: stable-implies3

STM: stable-implies4

STM: last-state-change

STM: last-state-change2

STM: last-state-change3

ABS: es-frame(es;i;L;x;T) **es-frame**

STM: es-frame_wf

STM: es-stable-1

STM: es-stable-2

STM: es-stable-3

STM: es-constant-1

ABS: es-responsive(es;l_1;tg_1;l_2;tg_2) **es-responsive**

STM: es-responsive_wf

STM: es-responsive-bijection

ABS: only $k(v)$:B sends [tg,$f(s;v)$] : T on l **es-only-sender**

STM: es-only-sender_wf

ABS: @i x has type T **es-type**

STM: vartype-es-type

STM: vartype-es-state-sub

STM: es-state-subtype

STM: state-after-pred-ds

STM: es-when-first

STM: init-p-implies

ABS: usends1-p(es;ds;k;T;l;tg;B;f) **usends1-p**

STM: usends1-p_wf

ABS: pre-init1-p(es;i;x;X;x_0;a;T;P) **pre-init1-p**

STM: pre-init1-p_wf

ABS: weak-precond-send-p(es;T;A;l;tg;a;ds;P;f) **weak-precond-send-p**

STM: weak-precond-send-p_wf

ABS: @i locl(a) occurs once **once-p**

STM: once-p_wf

ABS: locl(a) sends [tg,$f\{A\rightarrow T\}(x)$] on link l once **send-once-p**

STM: send-once-p_wf

STM: es-interval-induction

STM: es-interval-induction2

ABS: PossibleEvent($poss$) **possible-event**

STM: possible-event_wf

ABS: pe-es(e) **pe-es**

STM: pe-es_wf

ABS: pe-e(p) **pe-e**

STM: pe-e_wf

ABS: pe-state(p) **pe-state**

STM: pe-state_wf

ABS: pe-loc(p) **pe-loc**

STM: pe-loc_wf

ABS: K(P)@e **es-knows**

STM: es-knows_wf

STM: es-knows-true

STM: es-knows-knows

STM: es-knows-not

STM: es-knows-trans

STM: es-knows-valid

ABS: $e_1 \leq e_2$ **poss-le**

STM: poss-le_wf

STM: es-knows-stable

ABS: e:$s.P(s)$@j **es-simul**

STM: es-simul_wf

ABS: es-decls(es;i;ds;da) **es-decls**

STM: es-decls_wf

STM: es-decls-iff

STM: es-decls-join-single

ABS: with decls ds dasends on l from e include $f(e)$ and only these for tags in tgs

es-sends-iff

STM: es-sends-iff_wf

ABS: state dsk:A sends [tg, $e.f(e)$:B] on l **es-kind-sends-iff**

STM: es-kind-sends-iff_wf

STM: es-sends-iff_functionality

ABS: es-update-iff(es;i;x;ds;$e.P(e)$;$s.f(s)$) **es-update-iff**

STM: es-update-iff_wf

ABS: es-sends-iff2(es;l;tg;B;ds;$e.P(e)$;$e.f(e)$) **es-sends-iff2**

STM: es-sends-iff2_wf

STM: es-sends-iff2_functionality

ABS: event-info(ds;da) **event-info**

STM: event-info_wf

ABS: es-info(es;e) **es-info**

STM: es-info_wf

ABS: es-hist{i:l}(es;e_1;e_2) **es-hist**

STM: es-hist_wf

STM: member-es-hist

STM: null-es-hist

STM: es-hist-iseg

STM: es-hist-partition

STM: es-hist-last

STM: last-es-hist

STM: es-hist-is-append

STM: es-hist-is-concat

STM: iseg-es-hist

STM: es-hist-one-one

ABS: es-trans-state-from{i:l}(es;ks;g;z;e_1;e_2) **es-trans-state-from**

STM: es-trans-state-from_wf

ABS: e_2 = first $e \geq e_1.P(e)$ **es-first-since**

STM: es-first-since_wf

STM: es-first-since_functionality_wrt_iff

STM: alle-between1-not-first-since

STM: alle-between2-not-first-since

STM: es-increasing-sequence

STM: es-increasing-sequence2

ABS: [e_1;e_2]$\sim$([a,b].p(a;b))$*$[a,b].q(a;b) **es-pstar-q**

STM: es-pstar-q_wf

STM: es-pstar-q-trivial

STM: es-pstar-q-le

STM: es-pstar-q_functionality_wrt_implies

STM: es-pstar-q_functionality_wrt_rev_implies

STM: es-pstar-q_functionality_wrt_iff

STM: es-pstar-q-partition

ABS: $[e_1,e_2]\sim([a,b].p(a;b))+$ **es-pplus**

STM: es-pplus_wf

STM: es-pplus_functionality_wrt_implies

STM: es-pplus_functionality_wrt_rev_implies

STM: es-pplus_functionality_wrt_iff

STM: es-pplus-trivial

STM: es-pplus-le

STM: es-pplus-alle-between2

STM: es-pplus-partition

STM: es-pplus-first-since

STM: es-pplus-first-since-exit

ABS: data(T) **data**

STM: data_wf

ABS: secret-table(T) **secret-table**

STM: secret-table_wf

ABS: $\|tab\|$ **st-length**

STM: st-length_wf

ABS: ptr(tab) **st-ptr**

STM: st-ptr_wf

ABS: st-atom(tab;n) **st-atom**

STM: st-atom_wf

ABS: atoms-distinct(tab) **st-atoms-distinct**

STM: st-atoms-distinct_wf

ABS: next(tab) **st-next**

STM: st-next_wf

ABS: key(tab;n) **st-key**

STM: st-key_wf

ABS: data(tab;n) **st-data**

STM: st-data_wf

STM: st-ptr-wf2

ABS: st-lookup(tab;x) **st-lookup**

STM: st-lookup_wf

STM: st-lookup-property

STM: st-lookup-outl

STM: st-lookup-distinct

ABS: st-key-match(tab;k_1;k_2) **st-key-match**

STM: st-key-match_wf

ABS: decrypt(tab;$kval$) **st-decrypt**

STM: st-decrypt_wf

ABS: encrypt(tab;$keyv$) **st-encrypt**

STM: st-encrypt_wf

STM: st-length-encrypt

STM: st-atom-encrypt

ABS: ?[x] **cond-to-list**

STM: cond-to-list_wf

ABS: es-secret-server{\$table:ut2, \$encrypt:ut2, \$decrypt:ut2}
(es; T; L; i)

es-secret-server

STM: es-secret-server_wf

STM: ss-ptr-non-decreasing

STM: ss-table-length

STM: ss-atom-constant

STM: ss-atoms-distinct

STM: ss-encrypt-unique

ABS: es-seq(es;S) **es-seq**

STM: es-seq_wf

STM: send-minimal-lemma

STM: better-send-minimal-lemma

http://www.cs.cornell.edu/Info/Projects/NuPrl/FDLcontent/p0_286125_/p12_61797_{event-structure-theory}.html