message-automata[11,40]

ABS: Msg(da) **ma-Msg**

STM: ma-Msg_wf

ABS: MsgA **msga**

STM: msga_wf

ABS: ds(M) **ma_ds**

STM: ma_ds_wf

ABS: da(M) **ma_da**

STM: ma_da_wf

ABS: M.ds(x) **ma-ds**

STM: ma-ds_wf

ABS: M.da(a) **ma-da**

ABS: a declared in M **ma-decla**

STM: ma-decla_wf

ABS: rcv(l,tg) declared in M **ma-declm**

STM: ma-declm_wf

STM: ma-da_wf

ABS: M.din(l,tg) **ma-din**

STM: ma-din_wf

ABS: M.dout(l,tg) **ma-dout**

STM: ma-dout_wf

ABS: M.init(x,v) **ma-init**

STM: ma-init_wf

ABS: ma-init-const(M;x) **ma-init-const**

STM: ma-init-const_wf

ABS: M.init(x)?v **ma-init-val**

STM: ma-init-val_wf

ABS: M.state **ma-st**

STM: ma-st_wf

ABS: M.(timed)state **ma-tst**

STM: ma-tst_wf

ABS: read-state(s) **read-state**

STM: read-state_wf

STM: read-state_wf2

ABS: shift-state(s) **shift-state**

STM: shift-state_wf

STM: shift-state_wf2

ABS: M.Msg **ma-msg**

STM: ma-msg_wf

ABS: M.V(k) **ma-v**

STM: ma-v_wf

ABS: unsolvable M.pre(a,s) **ma-npre**

STM: ma-npre_wf

ABS: M.pre(a,s) **ma-pre**

STM: ma-pre_wf

ABS: a in dom(M.pre) **ma-has-pre**

STM: ma-has-pre_wf

STM: decidable__ma-has-pre

ABS: M.ef(k,x,s,v,w) **ma-ef**

STM: ma-ef_wf

ABS: ma-ef-const(M;k;x;s;v) **ma-ef-const**

STM: ma-ef-const_wf

ABS: M.ef(k,x,s,v)?w **ma-ef-val**

STM: ma-ef-val_wf

ABS: M.send(k;l;s;v;ms;i) **ma-send**

STM: ma-send_wf

ABS: M.sends(k,s,v) **ma-send-val**

STM: ma-send-val_wf

ABS: M sends on link l **ma-sends-on**

STM: ma-sends-on_wf

ABS: M.dout2(l;tg) **ma-dout2**

STM: ma-dout2_wf

ABS: M.frame(k affects x) **ma-frame**

STM: ma-frame_wf

ABS: M.sframe(k sends $<l,tg>$) **ma-sframe**

STM: ma-sframe_wf

ABS: M.aframe(k affects x) **ma-aframe**

STM: ma-aframe_wf

ABS: M.bframe(k sends on l) **ma-bframe**

STM: ma-bframe_wf

ABS: M.rframe(k reads x) **ma-rframe**

STM: ma-rframe_wf

ABS: M:k may not read x **ma-no-read**

STM: ma-no-read_wf

STM: assert-ma-no-read

ABS: ($s_1 \equiv s_2$ mod x) **ma-x-equiv**

STM: ma-x-equiv_wf

ABS: ma-x-tequiv(M;x;s_1;s_2) **ma-x-tequiv**

STM: ma-x-tequiv_wf

STM: ma-x-tequiv-shift-state

STM: ma-x-equiv-read-state

ABS: M.rframe(A.pre p for a) **ma-rframe-pre**

ABS: M.rframe(A.effect f of k on y) **ma-rframe-ef**

ABS: M.rframe(A.sends tfL of k on l) **ma-rframe-send**

ABS: ma-prob(M;b) **ma-prob**

STM: ma-prob_wf

ABS: $b \in$ dom(M.prob) **ma-prob-dom**

STM: ma-prob-dom_wf

ABS: ma-prob-da-dom(M;b) **ma-prob-da-dom**

STM: ma-prob-da-dom_wf

ABS: ma-random(M;T;v;i;a;n) **ma-random**

STM: ma-random_wf

ABS: $M_1 \subseteq M_2$ **ma-sub**

STM: ma-sub_wf

STM: ma-sub_weakening

ABS: ma{*ds*;
da;
init;
pre;
ef;
send;
frame;
sframe;
aframe;
bframe;
rframe;
prob}

mk-ma

STM: mk-ma_wf

ABS: **ma-empty**

STM: ma-empty_wf

ABS: M_1 $\|$decl M_2 **ma-compatible-decls**

ABS: $M_1 \oplus M_2$ **ma-join**

STM: ma-join_wf

STM: ma-join-sends-on

STM: ma-sub-join-left

ABS: $M_1 \parallel M_2$ **ma-compatible**

STM: ma-compatible_wf

STM: ma-compatible-symmetry

STM: ma-compatible-self

STM: ma-sub-join-right

STM: ma-empty-compatible-left

STM: ma-empty-compatible-right

ABS: x : t initially $x = v$ **ma-single-init**

STM: ma-single-init_wf

ABS: only members of L affect x :t **ma-single-frame**

STM: ma-single-frame_wf

ABS: only L sends on (l with tg) **ma-single-sframe**

STM: ma-single-sframe_wf

ABS: k affects only members of L **ma-single-aframe**

STM: ma-single-aframe_wf

ABS: k sends only on links in L **ma-single-bframe**

STM: ma-single-bframe_wf

ABS: only members of L read x **ma-single-rframe**

STM: ma-single-rframe_wf

ABS: with declarations ds:dsda:daeffect of k(v) is x := f s v **ma-single-effect**

STM: ma-single-effect_wf

ABS: with declarations ds:dsda:dak(v) sends f s v on link l **ma-single-sends**

STM: ma-single-sends_wf

ABS: (precondition a:Outcome(p) is P:State(ds) -> Bool) **ma-single-pre**

STM: ma-single-pre_wf

ABS: ma-frame-compat(A;B) **ma-frame-compat**

STM: ma-frame-compat_wf

STM: ma-join-frame-compat

STM: ma-join-frame-compat2

ABS: ma-frame-compatible(A;B) **ma-frame-compatible**

STM: ma-frame-compatible_wf

ABS: ma-prob-da(M) **ma-prob-da**

STM: ma-prob-da_wf

ABS: Feasible(M) **ma-feasible**

STM: ma-feasible_wf

STM: ma-feasible-ma-no-read

STM: ma-feasible-rframe-ef

STM: ma-feasible-rframe-send

STM: ma-empty-feasible

STM: ma-frame-compatible_symmetry

STM: ma-empty-frame-compatible-left

STM: ma-empty-frame-compatible-right

STM: ma-join-compatible

ABS: A ||+ B **ma-compat**

STM: ma-compat_wf

STM: ma-compat-symmetry

STM: ma-compat_weakening

STM: ma-empty-compat-left

STM: ma-empty-compat-right

STM: ma-join-feasible

STM: ma-compat-join

STM: ma-compat-join2

STM: ma-send-send-val

STM: ma-send-val-nil

STM: ma-send-val-nil2

STM: ma-feasible-bframe

STM: w-withlnk_wf_ma

STM: ma-init-const-join

STM: ma-ef-const-join