```
theory Dynamic_model_v6
imports Main
begin
subsection {* Security State Machine *}
locale SM =
fixes s0 :: 's
fixes step :: "'s Rightarrow> 'e Rightarrow> 's"
fixes domain :: "'e Rightarrow> ('d option)"
fixes vpeq :: "'s Rightarrow> 'd Rightarrow> 's Rightarrow> bool"  ("(_
sim> _ sim> _)")
fixes interferes :: "'d Rightarrow> 's Rightarrow> 'd Rightarrow> bool"
("(_ @ _ leadsto>_)")
assumes
vpeq_transitive_lemma : "forall> s t r d. (s sim> d sim> t) and> (t
sim> d sim> r) longrightarrow> (s sim> d sim> r)" and
vpeq_symmetric_lemma : "forall> s t d. (s sim> d sim> t) longrightarrow>
(t sim> d sim> s)" and
vpeq_reflexive_lemma : "forall> s d. (s sim> d sim> s)" and
interf_reflexive: "forall>d s. (d @ s leadsto> d)"
begin

definition non_interferes ::  "'d Rightarrow> 's Rightarrow> 'd Rightarrow>
bool" ("(_ @ _ setminus>leadsto> _)")
where "(u @ s setminus>leadsto> v) equiv> (u @ s leadsto> v)"

definition ivpeq :: "'s Rightarrow> 'd set Rightarrow> 's Rightarrow>
bool" ("(_ approx> _ approx> _)")
where "ivpeq s D t equiv> forall> d in> D. (s sim> d sim> t)"

primrec run :: "'s Rightarrow> 'e list Rightarrow> 's"
where run_Nil: "run s [] = s" |
run_Cons: "run s (a#as) = run (step s a) as "

definition reachable :: "'s Rightarrow> 's Rightarrow> bool" ("(_ hookrightarrow>
_)" [70,71] 60) where
"reachable s1 s2 equiv>  (exists>as. run s1 as = s2 )"

definition reachable0 :: "'s Rightarrow> bool"  where
"reachable0 s equiv> reachable s0 s"

declare non_interferes_def[cong] and ivpeq_def[cong] and reachable_def[cong]
and reachable0_def[cong] and run.simps(1)[cong] and run.simps(2)[cong]

lemma reachable_s0 : "reachable0 s0"
by (metis SM.reachable_def SM_axioms reachable0_def run.simps(1))
```

```
lemma reachable_self : "reachable s s"
using reachable_def run.simps(1) by fastforce

lemma reachable_step : "s’ = step s a Longrightarrow> reachable s s’"
proof-
assume a0: "s’ = step s a"
then have "s’ = run s [a]" by auto
then show ?thesis using reachable_def by blast
qed

lemma run_trans : "forall>C T V as bs. T = run C as and> V = run T
bs longrightarrow> V = run C (as@bs)"
proof -
{
fix T V as bs
have "forall>C. T = run C as and> V = run T bs longrightarrow> V =
run C (as@bs)"
proof(induct as)
case Nil show ?case by simp
next
case (Cons c cs)
assume a0: "forall>C. T = run C cs and> V = run T bs longrightarrow>
V = run C (cs @ bs)"
show ?case
proof-
{
fix C
have "T = run C (c # cs) and> V = run T bs longrightarrow> V = run
C ((c # cs) @ bs) "
proof
assume b0: "T = run C (c # cs) and> V = run T bs"
from b0 obtain C’ where b2: "C’ = step C c and> T = run C’ cs" by auto
with a0 b0 have "V = run C’ (cs@bs)" by blast
with b2 show "V = run C ((c # cs) @ bs)"
using append_Cons run_Cons by auto
qed
}
then show ?thesis by blast
qed
qed
}
then show ?thesis by auto
qed

lemma reachable_trans : "lbrakk>reachable C T; reachable T Vrbrakk>
```

```
Longrightarrow> reachable C V"
proof-
assume a0: "reachable C T"
assume a1: "reachable T V"
from a0 have "C = T or> (exists>as. T = run C as)" by auto
then show ?thesis
proof
assume b0: "C = T"
show ?thesis
proof -
from a1 have "T = V or> (exists>as. V = run T as)" by auto
then show ?thesis
proof
assume c0: "T=V"
with a0 show ?thesis by auto
next
assume c0: "(exists>as. V = run T as)"
then show ?thesis using a1 b0 by auto
qed
qed
next
assume b0: "exists>as. T = run C as"
show ?thesis
proof -
from a1 have "T = V or> (exists>as. V = run T as)" by auto
then show ?thesis
proof
assume c0: "T=V"
then show ?thesis using a0 by auto
next
assume c0: "(exists>as. V = run T as)"
from b0 obtain as where d0: "T = run C as" by auto
from c0 obtain bs where d1: "V = run T bs" by auto
then show ?thesis using d0  run_trans by fastforce
qed
qed
qed
qed

lemma reachableStep : "lbrakk>reachable0 C; C' = step C arbrakk> Longrightarrow>
reachable0 C'"
apply (simp add: reachable0_def)
using reachable_step reachable_trans by blast

lemma reachable0_reach : "lbrakk>reachable0 C; reachable C C'rbrakk>
Longrightarrow> reachable0 C'"
```

```
using  reachable_trans by fastforce

declare reachable_def[cong del] and reachable0_def[cong del]
end

subsection{* Information flow security properties *}

locale SM_enabled = SM s0 step domain vpeq interferes
for s0 :: 's and
step :: "'s Rightarrow> 'e Rightarrow> 's" and
domain :: "'e Rightarrow> ('d option)" and
vpeq :: "'s Rightarrow> 'd Rightarrow> 's Rightarrow> bool"  ("(_ sim>
_ sim> _)") and
interferes :: "'d Rightarrow> 's Rightarrow> 'd Rightarrow> bool" ("(_
@ _ leadsto>_)")
+
assumes enabled0: "forall>s a. reachable0 s longrightarrow> (exists>
s'. s' = step s a)"
and   policy_respect: "forall>v u s t. (s sim> u sim> t)
                              longrightarrow> (interferes v s u = interferes
v t u)"
begin

lemma enabled : "reachable0 s Longrightarrow> (exists> s'. s' = step
s a)"
using enabled0 by simp

primrec sources :: "'e list Rightarrow> 'd Rightarrow> 's Rightarrow>
'd set" where
sources_Nil:"sources [] d s = {d}" |
sources_Cons:"sources (a # as) d s = (Union>{sources as d (step s a)})
union>
                              {w . w = the (domain a) and> (exists>v
. interferes w s v and> vin>sources as d (step s a))}"
declare sources_Nil [simp del]
declare sources_Cons [simp del]



primrec ipurge :: "'e list Rightarrow> 'd Rightarrow> 's  Rightarrow>
'e list" where
ipurge_Nil:   "ipurge [] u s = []" |
ipurge_Cons:  "ipurge (a#as) u s = (if (the (domain a) in> (sources
(a#as) u s))
                                          then
                                              a # ipurge as u (step
s a)
                                          else
                                              ipurge as u (step s a)
                                          )"

definition observ_equivalence :: "'s Rightarrow> 'e list Rightarrow>
's Rightarrow>
          'e list Rightarrow> 'd Rightarrow> bool" ("(_ lhd> _ cong>
_ lhd> _ @ _)")
where "observ_equivalence s as t bs d equiv>
               ((run s as) sim> d sim> (run t bs))"
declare observ_equivalence_def[cong]

lemma observ_equiv_sym:
"(s lhd> as cong> t lhd> bs @ d) Longrightarrow> (t lhd> bs cong> s
lhd> as @ d)"
using observ_equivalence_def vpeq_symmetric_lemma by blast

lemma observ_equiv_trans:
"lbrakk>reachable0 t; (s lhd> as cong> t lhd> bs @ d); (t lhd> bs cong>
x lhd> cs @ d)rbrakk> Longrightarrow> (s lhd> as cong> x lhd> cs @
d)"
using observ_equivalence_def vpeq_transitive_lemma by blast

definition noninterference_r :: "bool"
where "noninterference_r equiv> forall>d as s. reachable0 s longrightarrow>
(s lhd> as cong> s lhd> (ipurge as d s) @ d)"

definition noninterference :: "bool"
where "noninterference equiv> forall>d as. (s0 lhd> as cong> s0 lhd>
(ipurge as d s0) @ d)"

definition weak_noninterference :: "bool"
where "weak_noninterference equiv> forall>d as bs. ipurge as d s0 =
ipurge bs d s0
                                                  longrightarrow> (s0
lhd> as cong> s0 lhd> bs @ d)"

definition weak_noninterference_r :: "bool"
where "weak_noninterference_r equiv> forall>d as bs s. reachable0 s
and> ipurge as d s = ipurge bs d s
                                                  longrightarrow> (s
lhd> as cong> s lhd> bs @ d)"

definition noninfluence::"bool"
where "noninfluence equiv> forall> d as s t. reachable0 s and> reachable0
```

```
t
                                and> (s approx> (sources as d s) approx>
t)
                                longrightarrow> (s lhd> as cong> t
lhd> (ipurge as d t) @ d)"

definition weak_noninfluence ::"bool"
where "weak_noninfluence equiv> forall> d as bs s t . reachable0 s
and> reachable0 t and> (s approx> (sources as d s) approx> t)
                                     and> ipurge as d t = ipurge bs
d t
                                     longrightarrow> (s lhd> as cong>
t lhd> bs @ d)"

definition weak_noninfluence2 ::"bool"
where "weak_noninfluence2 equiv> forall> d as bs s t . reachable0 s
and> reachable0 t and> (s approx> (sources as d s) approx> t)
                                     and> ipurge as d s = ipurge bs
d t
                                     longrightarrow> (s lhd> as cong>
t lhd> bs @ d)"

definition nonleakage :: "bool"
where "nonleakage equiv> forall>d as s t. reachable0 s and> reachable0
t
                          and> (s approx> (sources as d s) approx>
t) longrightarrow> (s lhd> as cong> t lhd> as @ d)"

declare noninterference_r_def[cong] and noninterference_def[cong] and
weak_noninterference_def[cong] and
weak_noninterference_r_def[cong] and noninfluence_def[cong] and
weak_noninfluence_def[cong] and weak_noninfluence2_def[cong] and nonleakage_def[cong]

subsection{* Unwinding conditions*}

definition step_consistent :: "bool" where
"step_consistent equiv>  forall>a d s t. reachable0 s and> reachable0
t and> (s sim> d sim> t) and>
                              (((the (domain a)) @ s leadsto> d) longrightarrow>
(s sim> (the (domain a)) sim> t))
                              longrightarrow> ((step s a) sim> d sim>
(step t a))"

definition weakly_step_consistent :: "bool" where
"weakly_step_consistent equiv>  forall>a d s t. reachable0 s and> reachable0
t and> (s sim> d sim> t) and>
```

```
                              ((the (domain a)) @ s leadsto> d) and>
(s sim> (the (domain a)) sim> t)
                              longrightarrow> ((step s a) sim> d sim>
(step t a))"

definition dynamic_local_respect :: "bool" where
"dynamic_local_respect equiv> forall>a d s. reachable0 s and> not>((the
(domain a)) @ s leadsto> d) longrightarrow> (s sim> d sim> (step s
a)) "

(*definition policy_respect :: "bool" where
        "policy_respect equiv> forall>v u s t. reachable0 s and> reachable0
t and> (s sim> u sim> t)
                              longrightarrow> (interferes v s u = interferes
v t u)"*)

declare step_consistent_def [cong] and weakly_step_consistent_def [cong]
and
dynamic_local_respect_def [cong]

lemma step_cons_impl_weak : "step_consistent Longrightarrow> weakly_step_consistent"
using step_consistent_def weakly_step_consistent_def by blast

definition lemma_local :: "bool" where
"lemma_local equiv> forall>s a as u. the (domain a) notin> sources
(a # as) u s longrightarrow> (s approx> (sources (a # as) u s)  approx>
(step s a))"

lemma weak_with_step_cons:
assumes p1: weakly_step_consistent
and   p2: dynamic_local_respect
shows   step_consistent
proof -
{
fix a d s t
have "reachable0 s and> reachable0 t and> (s sim> d sim> t) and>
               (((the (domain a)) @ s leadsto> d) longrightarrow> (s
sim> (the (domain a)) sim> t))
               longrightarrow> ((step s a) sim> d sim> (step t a))"
proof -
{
assume a0: "reachable0 s"
assume a1: "reachable0 t"
assume a2: "(s sim> d sim> t)"
assume a3: "(((the (domain a)) @ s leadsto> d) longrightarrow> (s sim>
(the (domain a)) sim> t))"
have "((step s a) sim> d sim> (step t a))"
proof (cases "((the (domain a)) @ s leadsto> d)")
assume b0: "((the (domain a)) @ s leadsto> d)"
have b1: "(s sim> (the (domain a)) sim> t)"
using b0 a3 by auto
have b2: "((step s a) sim> d sim> (step t a))"
using a0 a1 a2 b0 b1 p1 weakly_step_consistent_def by blast
then show ?thesis by auto
next
assume b0: "not>((the (domain a)) @ s leadsto> d)"
have b1: "not>((the (domain a)) @ t leadsto> d)"
using a0 a1 a2 b0 policy_respect by auto
have b2: "s sim> d sim> (step s a)"
using b0 p2 a0 by auto
have b3: "t sim> d sim> (step t a)"
using b1 p2 a1 by auto
have b4: "((step s a) sim> d sim> (step t a))"
using b2 b3 a2 vpeq_symmetric_lemma vpeq_transitive_lemma by blast
then show ?thesis by auto
qed
}
then show ?thesis by auto
qed
}
then show ?thesis by auto
qed

subsection{* Lemmas for the inference framework *}


lemma sources_refl:"reachable0 s Longrightarrow> u in> sources as u
s"
apply(induct as arbitrary: s)
apply(simp add: sources_Nil)
apply(simp add: sources_Cons)
using enabled reachableStep
by metis


lemma lemma_1_sub_1 : "lbrakk>reachable0 s ;
                       dynamic_local_respect;
                       the (domain a) notin> sources (a # as) u s;
                       (s approx> (sources (a # as) u s) approx> t)rbrakk>
                      Longrightarrow> (s approx> (sources as u (step
s a)) approx> (step s a))"
apply (simp add:dynamic_local_respect_def sources_Cons)
```

```
by blast

lemma lemma_1_sub_2 : "lbrakk>reachable0 s ;
                       reachable0 t ;
                       dynamic_local_respect;
                       the (domain a) notin> sources (a # as) u s;

                       (s approx> (sources (a # as) u s) approx> t)rbrakk>

                      Longrightarrow> (t approx> (sources as u (step
s a)) approx> (step t a))"
proof -
assume a1: "reachable0 s"
assume a2: "reachable0 t"
assume a3: dynamic_local_respect
assume a6: "the (domain a) notin> sources (a # as) u s"
assume a7: "(s approx> (sources (a # as) u s) approx> t)"
have b1: "forall>v. vin>sources as u (step s a) longrightarrow> not>interferes
(the (domain a)) s v"
using a6 sources_Cons by auto
have b2: "sources (a # as) u s = sources as u (step s a)"
using a6 sources_Cons by auto
have b3: "forall>v. vin>sources as u (step s a) longrightarrow> (s
sim> v sim> t)"
using a7 b2 ivpeq_def by blast
have b4: "forall>v. vin>sources as u (step s a) longrightarrow> not>interferes
(the (domain a)) t v"
using a1 a2 policy_respect b1 b3  by blast
have b5: "forall>v. vin>sources as u (step s a) longrightarrow> (t
sim> v sim> (step t a))"
using a2 a3 b4 by auto
then show ?thesis
using ivpeq_def by auto
qed

lemma lemma_1_sub_3 : "lbrakk>
                       the (domain a) notin> sources (a # as) u s;
                       (s approx> (sources (a # as) u s) approx> t)rbrakk>

                      Longrightarrow> (s approx> (sources as u (step
s a)) approx> t)"
apply (simp add:sources_Cons)
apply (simp add:sources_Cons)
done

lemma lemma_1_sub_4 : "lbrakk>(s approx> (sources as u (step s a))
approx> t);
                              (s approx> (sources as u (step s a)) approx>
(step s a));
                              (t approx> (sources as u (step s a)) approx>
(step t a)) rbrakk>
                       Longrightarrow> ((step s a) approx>(sources as
u (step s a)) approx> (step t a))"
by (meson ivpeq_def vpeq_symmetric_lemma vpeq_transitive_lemma)


lemma lemma_1 : "lbrakk>reachable0 s;
                       reachable0 t;
                       step_consistent;
                       dynamic_local_respect;
                       (s approx> (sources (a # as) u s) approx> t)rbrakk>
                       Longrightarrow> ((step s a) approx> (sources
as u (step s a)) approx> (step t a))"
apply (case_tac "the (domain a)in>sources (a # as) u s")
apply (simp add: step_consistent_def)
apply (simp add: sources_Cons)
proof -
assume a1: dynamic_local_respect
assume a4: "the (domain a) notin> sources (a # as) u s"
assume a5: "(s approx> (sources (a # as) u s) approx> t)"
assume b0: "reachable0 s"
assume b1: "reachable0 t"

have a6:"(s approx> (sources as u (step s a)) approx> t)"
using a1 policy_respect a4 a5 lemma_1_sub_3 by auto
then have a7: "(s approx> (sources as u (step s a)) approx> (step s
a))"
using b0 a1 policy_respect a4 a5 lemma_1_sub_1 by auto
then have a8: "(t approx> (sources as u (step s a)) approx> (step t
a))"
using b1 b0 a1 policy_respect a4 a5 lemma_1_sub_2 by auto
then show " ((step s a) approx>(sources as u (step s a)) approx> (step
t a))"
using a6 a7 lemma_1_sub_4 by blast
qed

lemma lemma_2 : "lbrakk>reachable0 s;
                       dynamic_local_respect;
                       the (domain a) notin> sources (a # as) u srbrakk>
                       Longrightarrow> (s approx> (sources as u (step
s a)) approx> (step s a))"
apply (simp add:dynamic_local_respect_def)
apply (simp add:sources_Cons)
by blast

lemma sources_eq1: "forall>s t as u. reachable0 s and>
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> (sources as u s) = (sources as
u t)"
proof -
{
fix as
have "forall>s t u. reachable0 s and>
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> (sources as u s) = (sources as
u t)"
proof(induct as)
case Nil then show ?case by (simp add: sources_Nil)
next
case (Cons b bs)
assume p0: "forall>s t u.((reachable0 s)
                                and> (reachable0 t)
                                and> step_consistent
                                and> dynamic_local_respect
                                and> (s approx> (sources bs u s) approx>
t)) longrightarrow>
                                  (sources bs u s) = (sources bs u
t)"
then show ?case
proof -
{
fix s t u
assume p1: "reachable0 s"
assume p2: "reachable0 t"
assume p3: step_consistent
assume p5: "dynamic_local_respect"
assume p9: "(s approx> (sources (b # bs) u s) approx> t)"
have a2: "((step s b) approx> (sources bs u (step s b)) approx> (step
t b))"
using lemma_1 p1 p2 p3 policy_respect p5 p9 by blast
have a3: "sources (b # bs) u s = sources (b # bs) u t"
proof (cases "the (domain b) in> (sources (b # bs) u s)")
assume b0: "the (domain b) in> (sources (b # bs) u s)"
have b1: "s sim> (the(domain b)) sim> t"
using b0 p9 by auto
have b3: "interferes (the (domain b)) s u = interferes (the (domain
b)) t u "
using p1 p2 policy_respect p9 sources_refl by fastforce
have b4: "(sources bs u (step s b)) = (sources bs u (step t b))"
using a2 p0 p1 p2 p3 p5 reachableStep by blast
have b5: "forall>v. vin>sources bs u (step s b)
                          longrightarrow> interferes (the (domain b))
s v = interferes (the (domain b)) t v "
using p1 p2 ivpeq_def policy_respect p9 sources_Cons by fastforce
then show "sources (b # bs) u s = sources (b # bs) u t"
using b4 b5 sources_Cons by auto
next
assume b0: "the (domain b) notin> (sources (b # bs) u s)"
have b1: "sources (b # bs) u s = sources bs u (step s b)"
using b0 sources_Cons by auto
have b2: "(sources bs u (step s b)) = (sources bs u (step t b))"
using a2 p0 p1 p2 p3 p5 reachableStep by blast
have b3: "forall>v. vin>sources bs u (step s b)longrightarrow>not>
interferes (the (domain b)) s v "
using b0 sources_Cons by auto
have b4: "forall>v. vin>sources bs u (step s b)longrightarrow>not>
interferes (the (domain b)) t v "
using b1 b3 p1 p2 p9 policy_respect by fastforce
have b5: "forall>v. vin>sources bs u (step t b)longrightarrow>not>
interferes (the (domain b)) t v "
by (simp add: b2 b4)
have b6: "the (domain b) notin> (sources (b # bs) u t)"
using b0 b2 b5 sources.simps(2) by auto
have b7: "sources (b # bs) u t = sources bs u (step t b)"
using b6 sources_Cons by auto
then show ?thesis
by (simp add: b1 b2)
qed
}
then show ?thesis by blast
qed
qed

}
then show ?thesis by blast
qed

lemma ipurge_eq: "forall>s t as u. reachable0 s and>
```

```
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> (ipurge as u s) = (ipurge as u
t)"
proof -
{
fix as
have "forall>s t u. reachable0 s and>
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> (ipurge as u s) = (ipurge as u
t)"
proof(induct as)
case Nil then show ?case by (simp add: sources_Nil)
next
case (Cons b bs)
assume p0: "forall>s t u.((reachable0 s)
                                and> (reachable0 t)
                                and> step_consistent
                                and> dynamic_local_respect
                                and> (s approx> (sources bs u s) approx>
t))

                                longrightarrow> (ipurge bs u s) = (ipurge
bs u t)"
then show ?case
proof -
{
fix s t u
assume p1: "reachable0 s"
assume p2: "reachable0 t"
assume p3: step_consistent
assume p5: "dynamic_local_respect"
assume p9: "(s approx> (sources (b # bs) u s) approx> t)"
have a1: "((step s b) approx> (sources bs u (step s b)) approx> (step
t b))"
using lemma_1 p1 p2 p3 p5 p9 by blast
have a2: "(ipurge bs u (step s b)) = (ipurge bs u (step t b))"
using a1 p0 p1 p2 p3 p5 p9 reachableStep by blast
have a3: "sources (b # bs) u s = sources (b # bs) u t"
using p1 p2 p3 p5 p9 sources_eq1 by blast
have a4: "ipurge (b # bs) u s = ipurge (b # bs) u t"
proof (cases "the (domain b) in> (sources (b # bs) u s)")
assume b0: "the (domain b) in> (sources (b # bs) u s)"
have b1: "s sim> (the(domain b)) sim> t"
using b0 p9 by auto
have b3: "the (domain b) in> (sources (b # bs) u t)"
using a3 b0 by auto
then show ?thesis
using a2 b0 ipurge_Cons by auto
next
assume b0: "the (domain b) notin> (sources (b # bs) u s)"
have b1: "sources (b # bs) u s = sources bs u (step s b)"
using b0 sources_Cons by auto
have b3: "forall>v. vin>sources bs u (step s b)longrightarrow>not>
interferes (the (domain b)) s v "
using b0 sources_Cons by auto
have b4: "forall>v. vin>sources bs u (step s b)longrightarrow>not>
interferes (the (domain b)) t v "
using b1 b3 p1 p2 p9 policy_respect by fastforce
have b5: "the (domain b) notin> (sources (b # bs) u t)"
using a3 b1 b4 interf_reflexive by auto
have b6: "ipurge (b # bs) u s = ipurge bs u (step s b)"
using b0 by auto
have b7: "ipurge (b # bs) u t = ipurge bs u (step t b)"
using b5 by auto
then show ?thesis
using b6 b7 a2 by auto
qed
}
then show ?thesis by blast
qed
qed
}
then show ?thesis by blast
qed

lemma non_influgence_lemma: "forall>s t as u. reachable0 s and>
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> ((s lhd> as cong> t lhd> (ipurge
as u t) @ u))"
proof -
{
fix as
have  "forall>s t u. reachable0 s and>
                    reachable0 t and>
                    step_consistent and>
                    dynamic_local_respect and>
                    (s approx> (sources as u s) approx> t)
                    longrightarrow> ((s lhd> as cong> t lhd> (ipurge
as u t) @ u))"
proof (induct as)
case Nil show ?case using sources_Nil by auto
next
case (Cons b bs)
assume p0: "forall>s t u.((reachable0 s)
                                and> (reachable0 t)
                                and> step_consistent
                                and> dynamic_local_respect
                                and> (s approx> (sources bs u s) approx>
t)) longrightarrow>
                                  ((s lhd> bs cong> t lhd> (ipurge
bs u t) @ u))"
then show ?case
proof -
{
fix s t u
assume p1: "reachable0 s"
assume p2: "reachable0 t"
assume p3: step_consistent
assume p4: dynamic_local_respect
assume p8: "(s approx> (sources (b # bs) u s) approx> t)"
have a1: "((step s b) approx> (sources bs u (step s b)) approx> (step
t b))"
using lemma_1 p1 p2 p3 p4 p8 by blast
have "s lhd> b # bs cong> t lhd> ipurge (b # bs) u t @ u"
proof (cases "the (domain b) in> sources (b # bs) u s")
assume b0: "the (domain b) in> sources (b # bs) u s"
have b1: "interferes (the (domain b)) s u = interferes (the (domain
b)) t u "
using p1 p2 policy_respect p8 sources_refl by fastforce
have b2: "forall>v. vin>sources bs u (step s b)
                          longrightarrow> interferes (the (domain b))
s v = interferes (the (domain b)) t v "
using p1 p2 ivpeq_def policy_respect p8 sources_Cons by fastforce
have b3: "ipurge (b # bs) u t = b # (ipurge bs u (step t b))"
by (metis b0 ipurge_Cons p1 p2 p3 p4 p8 sources_eq1)
have b4: "(((step s b) lhd> bs cong> (step t b) lhd> (ipurge bs u (step
t b)) @ u))"
using a1 p0 p1 p2 p3 p4 reachableStep by blast
show ?thesis
using b3 b4 by auto
next
assume b0: "the (domain b) notin> sources (b # bs) u s"
have b1: "ipurge (b # bs) u t = (ipurge bs u (step t b))"
by (metis a1 b0 ipurge_Cons ipurge_eq p1 p2 p3 p4 p8 reachableStep)
have b2: "(s approx> (sources bs u (step s b)) approx> (step s b))"
using b0 lemma_2 p1 p4 by blast
have b3:"(s approx> (sources bs u (step s b)) approx> t)"
using b0 lemma_1_sub_3 p8 by blast
have b4: "((step s b) approx> (sources bs u (step s b)) approx> t)"
by (meson b3 b2 ivpeq_def vpeq_symmetric_lemma vpeq_transitive_lemma)
have b5: "(((step s b) lhd> bs cong> t lhd> (ipurge bs u t) @ u))"
using b4 p0 p1 p2 p3 p4 reachableStep by blast
have b6: "(t approx> (sources bs u (step s b)) approx> (step t b))"
using p1 p2 b0 lemma_1_sub_2 p4 p8 by blast
have b7: "ipurge bs u t = ipurge bs u (step t b)"
by (metis a1 b4 ipurge_eq p1 p2 p3 p4 reachableStep)
have b8: "(((step s b) lhd> bs cong> t lhd> (ipurge bs u (step t b))
@ u))"
using b5 b7 by auto
then show ?thesis
using b1 observ_equivalence_def run_Cons by auto
qed
}
then show ?thesis by blast
qed
qed
}
then show ?thesis by blast
qed

subsection{* Interference framework of information flow security properties
*}

theorem nonintf_impl_weak: "noninterference Longrightarrow> weak_noninterference"
by (metis noninterference_def observ_equiv_sym observ_equiv_trans reachable_s0
weak_noninterference_def)

theorem wk_nonintf_r_impl_wk_nonintf: "weak_noninterference_r Longrightarrow>
weak_noninterference"
using reachable_s0 by auto

theorem nonintf_r_impl_noninterf: "noninterference_r Longrightarrow>
noninterference"
using noninterference_def noninterference_r_def reachable_s0 by auto
```

```
theorem nonintf_r_impl_wk_nonintf_r: "noninterference_r Longrightarrow>
weak_noninterference_r"
by (metis noninterference_r_def observ_equiv_sym observ_equiv_trans
weak_noninterference_r_def)

lemma noninf_impl_nonintf_r: "noninfluence Longrightarrow> noninterference_r"
using ivpeq_def noninfluence_def noninterference_r_def vpeq_reflexive_lemma
by blast

lemma noninf_impl_nonlk: "noninfluence Longrightarrow> nonleakage"
using noninterference_r_def nonleakage_def observ_equiv_sym
observ_equiv_trans noninfluence_def noninf_impl_nonintf_r by blast


lemma wk_noninfl_impl_nonlk: "weak_noninfluence Longrightarrow> nonleakage"
using weak_noninfluence_def nonleakage_def by blast

lemma wk_noninfl_impl_wk_nonintf_r: "weak_noninfluence Longrightarrow>
weak_noninterference_r"
using ivpeq_def weak_noninfluence_def vpeq_reflexive_lemma weak_noninterference_r_def
by blast

lemma sources_step2:
"lbrakk>reachable0 s; (the (domain a))@s leadsto> drbrakk> Longrightarrow>
sources [a] d s = {the (domain a),d}"
apply(auto simp: sources_Cons sources_Nil enabled dest: enabled)
done

lemma exec_equiv_both:
"lbrakk>reachable0 C1; reachable0 C2;(step C1 a) lhd> as cong> (step
C2 b) lhd> bs @ urbrakk>
      Longrightarrow> (C1 lhd> (a # as) cong> C2 lhd> (b # bs) @ u)"
by auto

lemma sources_unwinding_step:
"lbrakk>reachable0 s; reachable0 t;s approx>(sources (a#as) d s)approx>
t; step_consistentrbrakk>
        Longrightarrow> ((step s a) approx>(sources as d (step s a))approx>
(step t a))"
apply(clarsimp simp: ivpeq_def sources_Cons)
using UnionI step_consistent_def by blast

lemma nonlk_imp_sc: "nonleakage Longrightarrow> step_consistent"
proof -
assume p0: "nonleakage"
have p1: "forall>as d s t. reachable0 s and> reachable0 t
```

```
                  and> (s approx> (sources as d s) approx> t) longrightarrow>
(s lhd> as cong> t lhd> as @ d)"
using p0 nonleakage_def by auto
have p2: "forall>a d s t. reachable0 s and> reachable0 t and> (s sim>
d sim> t) and>
                 (((the (domain a)) @ s leadsto> d) longrightarrow>
(s sim> (the (domain a)) sim> t))
                 longrightarrow> ((step s a) sim> d sim> (step t a))"
proof -
{
fix a d s t
assume a0: "reachable0 s and> reachable0 t and> (s sim> d sim> t) and>

                 (((the (domain a)) @ s leadsto> d) longrightarrow>
(s sim> (the (domain a)) sim> t))"
have a4: "s approx> (sources [] d s) approx> t"
using a0 sources_Nil by auto
have a5: "(s lhd> [] cong> t lhd> [] @ d)"
using a4 a0 p1 by auto
have a6: "((step s a) sim> d sim> (step t a))"
proof (cases "(the (domain a))@s leadsto> d")
assume b0: "(the (domain a))@s leadsto> d"
have b1: "sources [a] d s = {d, (the(domain a))}"
using b0 sources_Cons sources_Nil by auto
have c0: "(s sim> (the (domain a)) sim> t)"
using b0 a0 by auto
have b2: "s approx> (sources [a] d s) approx> t"
using b1 a0 c0 by auto
have b3: "(s lhd> [a] cong> t lhd> [a] @ d)"
using b2 a0 p1 by auto
have b4: "((step s a) sim> d sim> (step t a))"
using b3 by auto
then show ?thesis by auto
next
assume b0: "not>((the (domain a))@s leadsto> d)"
have b1: "sources [a] d s = {d}"
using b0 sources_Cons sources_Nil by auto
have b2: "(s approx> (sources [a] d s) approx> t)"
using b1 a0 by auto
have b3: "(s lhd> [a] cong> t lhd> [a] @ d)"
using b2 a0 p1 by auto
have b4: "((step s a) sim> d sim> (step t a))"
using b3 by auto
then show ?thesis by auto
qed
}
```

```
then show ?thesis
by auto
qed
then show ?thesis by auto
qed

lemma sc_imp_nonlk: "step_consistent Longrightarrow> nonleakage"
proof -
assume p0: "step_consistent"
have p1: "forall>a d s t. reachable0 s and> reachable0 t and> (s sim>
d sim> t) and>
               (s sim> (the (domain a)) sim> t) longrightarrow> ((step
s a) sim> d sim> (step t a))"
using p0 step_consistent_def by auto
have p2: "forall>as d s t. reachable0 s and> reachable0 t
                  and> (s approx> (sources as d s) approx> t) longrightarrow>
(s lhd> as cong> t lhd> as @ d)"
proof -
{
fix as
have "forall>d s t. reachable0 s and> reachable0 t
                  and> (s approx> (sources as d s) approx> t) longrightarrow>
(s lhd> as cong> t lhd> as @ d)"
proof (induct as)
case Nil show ?case using sources_refl by auto
next
case (Cons b bs)
assume a0: "forall>d s t. reachable0 s and> reachable0 t
                  and> (s approx> (sources bs d s) approx> t) longrightarrow>
(s lhd> bs cong> t lhd> bs @ d)"
show ?case
proof -
{
fix d s t
assume b0: "reachable0 s and> reachable0 t"
assume b1: "(s approx> (sources (b#bs) d s) approx> t)"
have b2: "((step s b) approx>(sources bs d (step s b))approx> (step
t b))"
using b0 b1 p0 sources_unwinding_step by auto
have b3: "(step s b) lhd> bs cong> (step t b) lhd> bs @ d"
using Cons.hyps b0 b2 reachableStep by blast
have b4: "s lhd> b # bs cong> t lhd> b # bs @ d"
using b3 by auto
}
then show ?thesis by auto
qed
qed
}
then show ?thesis by auto
qed
then show ?thesis by auto
qed

theorem sc_eq_nonlk: "step_consistent = nonleakage"
using nonlk_imp_sc sc_imp_nonlk by blast

lemma noninf_imp_dlr: "noninfluence Longrightarrow> dynamic_local_respect"
proof -
assume p0: "noninfluence"
have p1: "forall> d as s t. reachable0 s and> reachable0 t
                  and> (s approx> (sources as d s) approx> t)
                  longrightarrow> (s lhd> as cong> t lhd> (ipurge as
d t) @ d)"
using p0 noninfluence_def by auto
have "forall>a d s. reachable0 s and> not>((the (domain a)) @ s leadsto>
d)
                    longrightarrow> (s sim> d sim> (step s a)) "
proof -
{
fix a d s
assume a0: "reachable0 s and> not>((the (domain a)) @ s leadsto> d)"
have a1: "sources [a] d s = {d}"
using a0 sources_Cons sources_Nil by auto
have a2: "(ipurge [a] d s) = []"
using a0 a1 interf_reflexive by auto
have a3: "s sim> d sim> s"
using vpeq_reflexive_lemma by auto
have a4: "(s approx> (sources [a] d s) approx> s)"
using a1 a3 by auto
have a5: "(s lhd> [a] cong> s lhd> (ipurge [a] d s) @ d)"
using a4 a0 p1 by auto
have a6: "(s lhd> [a] cong> s lhd> [] @ d)"
using a5 a2 by auto
have a7: "(s sim> d sim> (step s a))"
using a6 vpeq_symmetric_lemma by auto
}
then show ?thesis by auto
qed
then show ?thesis by auto
qed

lemma noninf_imp_sc: "noninfluence Longrightarrow> step_consistent"
```

```
using nonlk_imp_sc noninf_impl_nonlk by blast

theorem UnwindingTheorem : "lbrakk>step_consistent;
                            dynamic_local_respectrbrakk>
                            Longrightarrow> noninfluence"
proof -
assume p3: step_consistent
assume p4: dynamic_local_respect
{
fix as d
have "forall>s t. reachable0 s and>
                  reachable0 t and>
                  (s approx> (sources as d s) approx> t)
                  longrightarrow> ((s lhd> as cong> t lhd> (ipurge
as d  t) @ d))"
proof(induct as)
case Nil show ?case using sources_Nil by auto
next
case (Cons b bs)
assume p0: "forall>s t. reachable0 s and>
                  reachable0 t and>
                  (s approx> (sources bs d s) approx> t)
                  longrightarrow> ((s lhd> bs cong> t lhd> (ipurge
bs d  t) @ d))"
then show ?case
proof -
{
fix s t
assume p1: "reachable0 s"
assume p2: "reachable0 t"
assume p8: "(s approx> (sources (b # bs) d s) approx> t)"
have a1: "((step s b) approx> (sources bs d (step s b)) approx> (step
t b))"
using lemma_1 p1 p2 p3 p4 p8 by blast
have a2: "s lhd> b # bs cong> t lhd> ipurge (b # bs) d t @ d"
proof (cases "the (domain b) in> sources (b # bs) d s")
assume b0: "the (domain b) in> sources (b # bs) d s"
have b1: "interferes (the (domain b)) s d = interferes (the (domain
b)) t d "
using p1 p2 policy_respect p8 sources_refl by fastforce
have b2: "forall>v. vin>sources bs d (step s b)
                           longrightarrow> interferes (the (domain b))
s v = interferes (the (domain b)) t v "
using p1 p2 ivpeq_def policy_respect p8 sources_Cons by fastforce
have b3: "ipurge (b # bs) d t = b # (ipurge bs d (step t b))"
by (metis b0 ipurge_Cons p1 p2 p3 p4 p8 sources_eq1)
have b4: "(((step s b) lhd> bs cong> (step t b) lhd> (ipurge bs d (step
t b)) @ d))"
using a1 p0 p1 p2 p3 p4 reachableStep by blast
then show ?thesis
using b3 b4 by auto
next
assume b0: "the (domain b) notin> sources (b # bs) d s"
have b1: "ipurge (b # bs) d t = (ipurge bs d (step t b))"
by (metis a1 b0 ipurge_Cons ipurge_eq p1 p2 p3 p4 p8 reachableStep)
have b2: "(s approx> (sources bs d (step s b)) approx> (step s b))"
using b0 lemma_2 p1 p4 by blast
have b3:"(s approx> (sources bs d (step s b)) approx> t)"
using b0 lemma_1_sub_3 p8 by blast
have b4: "((step s b) approx> (sources bs d (step s b)) approx> t)"
by (meson b3 b2 ivpeq_def vpeq_symmetric_lemma vpeq_transitive_lemma)
have b5: "(((step s b) lhd> bs cong> t lhd> (ipurge bs d t) @ d))"
using b4 p0 p1 p2 p3 p4 reachableStep by blast
have b6: "(t approx> (sources bs d (step s b)) approx> (step t b))"
using p1 p2 b0 lemma_1_sub_2 p4 p8 by blast
have b7: "ipurge bs d t = ipurge bs d (step t b)"
by (metis a1 b4 ipurge_eq p1 p2 p3 p4 reachableStep)
have b8: "(((step s b) lhd> bs cong> t lhd> (ipurge bs d (step t b))
@ d))"
using b5 b7 by auto
then show ?thesis
using b1 observ_equivalence_def run_Cons by auto
qed
}
then show ?thesis by blast
qed
qed
}
then show ?thesis using noninfluence_def by blast
qed

theorem UnwindingTheorem1 : "lbrakk>weakly_step_consistent;
                              dynamic_local_respectrbrakk>  Longrightarrow>
noninfluence"
using UnwindingTheorem weak_with_step_cons by blast

theorem uc_eq_noninf : "(step_consistent and> dynamic_local_respect)
= noninfluence"
using UnwindingTheorem1 step_cons_impl_weak noninf_imp_dlr noninf_imp_sc
by blast

(*
     dipurge_eq required
   *)

theorem noninf_impl_weak:"noninfluence Longrightarrow> weak_noninfluence"
proof -
assume p0: "noninfluence"
have p1: "forall> d as s t. reachable0 s and> reachable0 t
               and> (s approx> (sources as d s) approx> t)
               longrightarrow> (s lhd> as cong> t lhd> (ipurge as d
t) @ d)"
using p0 noninfluence_def by auto
have p2: "(step_consistent and> dynamic_local_respect)"
using p0 uc_eq_noninf by auto
have "forall> d as bs s t . reachable0 s and> reachable0 t and> (s
approx> (sources as d s) approx> t)
               and> ipurge as d t = ipurge bs d t
               longrightarrow> (s lhd> as cong> t lhd> bs @ d)"
proof -
{
fix d as bs s t
assume a0: "reachable0 s and> reachable0 t and> (s approx> (sources
as d s) approx> t)
               and> ipurge as d t = ipurge bs d t"
have a4: "noninterference_r"
using noninf_impl_nonintf_r p0 by auto
have a7: "weak_noninterference_r"
using a4 nonintf_r_impl_wk_nonintf_r by auto
have a6: "ipurge as d s = ipurge as d t"
using a0 p2 ipurge_eq by auto
have b1: "(s lhd> as cong> t lhd> (ipurge as d t) @ d)"
using a0 p1 by auto
have b4: "(s lhd> as cong> t lhd> as @ d)"
using a0 noninf_imp_sc nonleakage_def p0 sc_imp_nonlk by blast
have b5: "(t lhd> bs cong> t lhd> (ipurge bs d t) @ d)"
using a0 a4 by auto
have b6: "(t lhd> bs cong> t lhd> (ipurge as d t) @ d)"
using b5 a0 by auto
have b7: "(s lhd> as cong> t lhd> bs @ d)"
using a0 b1 b6 observ_equiv_sym observ_equiv_trans by blast
}
then show ?thesis by auto
qed
then show ?thesis by auto
qed
```

```
lemma wk_nonintf_r_and_nonlk_impl_noninfl: "lbrakk>weak_noninterference_r;
nonleakagerbrakk> Longrightarrow> weak_noninfluence"
proof -
assume p0: "weak_noninterference_r"
assume p1: "nonleakage"
have p2: "forall>d as bs s. reachable0 s and> ipurge as d s = ipurge
bs d s
                              longrightarrow> (s lhd> as cong> s lhd>
bs @ d)"
using weak_noninterference_r_def p0 by auto
have p3: "forall>d as s t. reachable0 s and> reachable0 t
                          and> (s approx> (sources as d s) approx>
t) longrightarrow> (s lhd> as cong> t lhd> as @ d)"
using nonleakage_def p1 by auto
have "forall> d as bs s t . reachable0 s and> reachable0 t and> (s
approx> (sources as d s) approx> t)
                              and> ipurge as d t = ipurge bs d t
                              longrightarrow> (s lhd> as cong> t lhd>
bs @ d)"
proof -
{
fix d as bs s t
assume a0: "reachable0 s and> reachable0 t and> (s approx> (sources
as d s) approx> t)
                              and> ipurge as d t = ipurge bs d t"
have a1: "s lhd> as cong> t lhd> as @ d"
using a0 p3 by blast
have a2: "t lhd> as cong> t lhd> bs @ d"
using a0 p2 by auto
have a3: "(s lhd> as cong> t lhd> bs @ d)"
using a0 a1 a2 observ_equiv_trans by blast
}
then show ?thesis by auto
qed
then show ?thesis by auto
qed

lemma nonintf_r_and_nonlk_impl_noninfl: "lbrakk>noninterference_r;
nonleakagerbrakk> Longrightarrow> noninfluence"
proof -
assume p0: "noninterference_r"
assume p1: "nonleakage"
have p2: " forall>d as s. reachable0 s longrightarrow> (s lhd> as cong>
s lhd> (ipurge as d s) @ d)"
using p0 noninterference_r_def by auto
have p3: "forall>d as s t. reachable0 s and> reachable0 t
```

```
                          and> (s approx> (sources as d s) approx>
t) longrightarrow> (s lhd> as cong> t lhd> as @ d)"
using p1 nonleakage_def by auto
have "forall> d as s t. reachable0 s and> reachable0 t
                          and> (s approx> (sources as d s) approx>
t)
                          longrightarrow> (s lhd> as cong> t lhd> (ipurge
as d t) @ d)"
proof -
{
fix d as bs s t
assume a0: "reachable0 s and> reachable0 t
                          and> (s approx> (sources as d s) approx>
t)"
have a1: "s lhd> as cong> t lhd> as @ d"
using p3 a0 by blast
have a2: "s lhd> as cong> s lhd> (ipurge as d s) @ d"
using a0 p2 by fast
have a3: "t lhd> as cong> t lhd> (ipurge as d t) @ d"
using a0 p2 by fast
have "s lhd> as cong> t lhd> (ipurge as d t) @ d"
using a0 a1 a3 observ_equiv_trans by blast
}
then show ?thesis by auto
qed
then show ?thesis using noninfluence_def by blast
qed

theorem nonintf_r_and_nonlk_eq_strnoninfl: "(noninterference_r and>
nonleakage) = noninfluence"
using nonintf_r_and_nonlk_impl_noninfl noninf_impl_nonintf_r noninf_impl_nonlk
by blast

end
end
```