

Interactive Verification Using Averest

Abstract—This reference card lists the functions of module `Averest.Analysis.ProofTactics` which implement an interactive verification framework for Quartz programs. To use it, you first need to create a `ProofContext` which is invisible by the user by loading a Quartz module using one of the provided parse functions. You can then set a new proof goal $\Gamma \vdash \varphi$ with a name, an assumption list Γ and conclusion φ using the function `SetNewGoal`. This proof goal is then the root goal of the subgoals that are generated during its proof by tactic applications. Proving a goal involves applying a sequence of tactics, each of which extend the proof tree by adding the generated subgoals as child nodes of the current proof goal. If a tactic does not generate new subgoals, the current proof goal is closed (proven), and the next open proof goal of the same root goal is considered as the new current proof goal. If there are no further open proof goals, the root goal becomes a theorem which can then be accessed with its name in other proofs. Note that there are initial goals which refer to the initial point of time, and others which refer to any point of time. The meaning of a goal is that it holds for all possible computations of the Quartz module, i.e., for all initial states or all initial paths, respectively.

PROOF TREE CONTEXT MANAGEMENT

Loading Quartz Modules

- `LoadModuleFromStream(inStream:TextReader)`
- `LoadModuleFromString(s:string)`
- `LoadModuleFromFile(filename:string)`
- `LoadModuleFromInternet(url:string)`
- `ResetProofContext`

Parsing Functions

- `ParseSpecExprFromString(s:string)`
- `ParseExprFromString(s:string)`

Proof Goal Management

- `RemoveCurrGoal`
- `SetNewGoal name asm concl`
- `ProveThm name phi tactic`
- `ProveThmAsm name asm phi tactic`

Printing Functions

- `PrintInitEquations`
- `PrintNowEquations`
- `PrintNxtEquations`
- `PrintCurrGoal`
- `PrintOpenGoals`
- `PrintTheorems`
- `PrintCurrRootGoal`
- `PrintProofOfThm(thmName:string)`
- `PrintProofTreeOfCurrGoal`
- `PrintTacticOfNode(i:int)`
- `PrintTacticOfThm(thmName:string)`

PROOF TACTICS

Proof Tactics

- `AcceptTac` convert the root goal to a theorem without proof (should only be used for a first proof sketch)
- `AsmDropTac(i)` remove assumption i :

$$\frac{\gamma_0, \dots, \gamma_{n-1} \vdash \varphi}{\gamma_0, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}$$

- `AsmDropListTac[i1;...;in]` remove the listed assumptions
- `AsmConjTac(i)`

$$\frac{\gamma_0, \dots, \gamma_{i-1}, \varphi \wedge \psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \Phi}{\gamma_0, \dots, \gamma_{i-1}, \varphi, \psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \Phi}$$

- `AsmDisjTac(i)`

$$\frac{\gamma_0, \dots, \gamma_{i-1}, \varphi \vee \psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \Phi}{\gamma_0, \dots, \gamma_{i-1}, \varphi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \Phi}$$

$$\gamma_0, \dots, \gamma_{i-1}, \psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \Phi$$

- `AsmInstTac(i)`

$$\frac{\gamma_0, \dots, \gamma_{i-1}, G\psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}{\psi, \gamma_0, \dots, \gamma_{i-1}, G\psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}$$

- `AsmInstNextTac n i`

$$\frac{\gamma_0, \dots, \gamma_{i-1}, G\psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}{X^n\psi, \gamma_0, \dots, \gamma_{i-1}, G\psi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}$$

- `AsmMpTac(i)`

$$\frac{\gamma_0, \dots, \gamma_{i-1}, \psi \rightarrow \varphi, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \varphi}{\gamma_0, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \psi}$$

- `AsmRewriteTac` scans the assumptions for literals (possibly negated variables) and replaces them in the goal's conclusion by either true or false
- `AsmSplitTac` converts all assumptions into DNF and makes then case splits on all cubes to prepare `AsmRewriteTac`
- `AssumeThmTac(thmName:string)` add the listed theorem to the assumptions
- `AssumeInstThmTac(thmName:string)` similar to tactic `AssumeThmTac`, this tactic adds a theorem to the assumptions after removing universal quantifiers, i.e., for theorem $\vdash G\psi$ the goal $\Gamma \vdash \varphi$ becomes $\psi, \Gamma \vdash \varphi$
- `AssumeNextInstThmTac(thmName:string)` similar to tactic `AssumeThmTac`, this tactic adds an instantiation for the next point in time of a theorem to the assumptions, i.e., for theorem $\vdash G\psi$ the goal $\Gamma \vdash \varphi$ becomes $X\psi, \Gamma \vdash \varphi$
- `AxiomTac` converts trivial proof goals to theorems.

- BoolCasesTac(ψ)

$$\frac{\Gamma \vdash \varphi}{\neg\psi, \Gamma \vdash \varphi \quad \psi, \Gamma \vdash \varphi}$$

- BoolListCasesTac(psiL) makes a case distinction on the given list of expressions which generates an exponential number of subgoals!
- BootTac replace all control-flow labels and output variables with their initial values provided that the goal is an initial goal.
- ConjTac

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}$$

- ConjLeftTac

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi \quad \varphi, \Gamma \vdash \psi}$$

- ConjRightTac

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi \quad \psi, \Gamma \vdash \varphi}$$

- ContraposTac

$$\frac{\Gamma \vdash \varphi \rightarrow \psi}{\Gamma \vdash \neg\psi \rightarrow \neg\varphi}$$

- CutTac(ψ)

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi \quad \psi, \Gamma \vdash \varphi}$$

- DischTac

$$\frac{\Gamma \vdash \varphi \rightarrow \psi}{\varphi, \Gamma \vdash \psi} \quad \frac{\Gamma \vdash \neg\varphi}{\varphi, \Gamma \vdash \text{false}}$$

- DisjLeftTac

$$\frac{\Gamma \vdash \varphi \vee \psi}{\neg\psi, \Gamma \vdash \varphi}$$

- DisjRightTac

$$\frac{\Gamma \vdash \varphi \vee \psi}{\neg\varphi, \Gamma \vdash \psi}$$

- ExpandAllNowEquTac expand the current goal's conclusion with all immediate assignments
- ExpandNowEquTac(vName) expand the current goal's conclusion with the immediate assignment whose left hand side variable's name is vName
- ExpandAllNxtEquTac expand the current goal's conclusion with all next assignments
- ExpandNxtEquTac(vName) expand the current goal's conclusion with the next assignment whose left hand side variable's name is vName
- ExpandAllEquTac expand the current goal's conclusion with all assignments until there are no further changes (caution: this may not terminate)
- FairnessTac(χ)

$$\frac{\Gamma \vdash \text{GF}\varphi}{\Gamma \vdash \text{F}\varphi \quad \Gamma \vdash \text{G}(\varphi \rightarrow \text{X}\text{F}\varphi)}$$

- GenTac

$$\frac{\Gamma \vdash \text{A}\varphi}{\Gamma \vdash \varphi} \quad \frac{\Gamma \vdash \text{G}\varphi}{\Gamma \vdash \varphi}$$

- IffTac

$$\frac{\Gamma \vdash \varphi \leftrightarrow \psi}{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \psi \rightarrow \varphi}$$

- ImpResTac searches implications and equivalences in the assumption list and derives new assumptions as follows

$$\frac{\varphi, \varphi \rightarrow \psi, \Gamma \vdash \gamma}{\varphi, \psi, \Gamma \vdash \gamma} \quad \frac{\varphi, \varphi \leftrightarrow \psi, \Gamma \vdash \gamma}{\varphi, \psi, \Gamma \vdash \gamma}$$

- InductTac

$$\frac{\Gamma \vdash \text{G}\varphi}{\Gamma \vdash \varphi \quad \Gamma \vdash \text{G}(\varphi \rightarrow \text{X}\varphi)}$$

- InductWeakBeforeTac

$$\frac{\Gamma \vdash [\varphi \text{ B } \psi]}{\Gamma \vdash \neg\psi \quad \Gamma \vdash \text{G}(\neg\varphi \wedge \neg\psi \rightarrow \text{X}\neg\psi)}$$

- InductWeakUntilTac

$$\frac{\Gamma \vdash [\varphi \text{ U } \psi]}{\Gamma \vdash \varphi \vee \psi \quad \Gamma \vdash \text{G}(\varphi \wedge \neg\psi \rightarrow \text{X}(\varphi \vee \psi))}$$

- InductWeakWhenTac

$$\frac{\Gamma \vdash [\varphi \text{ W } \psi]}{\Gamma \vdash \psi \rightarrow \varphi \quad \Gamma \vdash \text{G}(\neg\psi \rightarrow \text{X}(\psi \rightarrow \varphi))}$$

- InductInvTac(χ)

$$\frac{\Gamma \vdash \text{G}\varphi}{\Gamma \vdash \chi \quad \Gamma \vdash \text{G}(\chi \rightarrow \varphi \wedge \text{X}\chi)}$$

- InductWeakBeforeInvTac(χ)

$$\frac{\Gamma \vdash [\varphi \text{ B } \psi]}{\Gamma \vdash \chi \quad \Gamma \vdash \text{G}((\varphi \rightarrow \psi) \wedge \chi \rightarrow \neg\psi \wedge \text{X}\chi)}$$

- InductWeakUntilInvTac(χ)

$$\frac{\Gamma \vdash [\varphi \text{ U } \psi]}{\Gamma \vdash \chi \quad \Gamma \vdash \text{G}(\neg\psi \wedge \chi \rightarrow \varphi \wedge \text{X}\chi)}$$

- InductWeakWhenInvTac(χ)

$$\frac{\Gamma \vdash [\varphi \text{ W } \psi]}{\Gamma \vdash \chi \quad \Gamma \vdash \text{G}(\neg(\varphi \wedge \psi) \wedge \chi \rightarrow \neg\psi \wedge \text{X}\chi)}$$

- LivenessBoundTac(bound: **int**) unroll the liveness property for bounded liveness checking

$$\frac{\Gamma \vdash \text{F}\psi}{\Gamma \vdash \psi \vee \text{X}(\psi \vee \text{X}(\dots))}$$

- LivenessNextTac prove the liveness property if it holds at the next point of time:

$$\frac{\Gamma \vdash \text{F}\psi}{\Gamma \vdash \text{X}\psi}$$

- LivenessRankTac(ϱ)

$$\frac{\Gamma \vdash \text{F}\psi}{\Gamma \vdash [(\text{next}(\varrho) < \varrho) \text{ U } \psi]}$$

- **LivenessRankInvTac**(ϱ, χ)

$$\frac{\Gamma \vdash F\psi}{\Gamma \vdash \chi \quad \Gamma \vdash G(\chi \wedge \neg\psi \rightarrow \text{next}(\varrho) < \varrho \wedge X\chi)}$$

- **LivenessTriggerRankInvTac**(σ, ϱ, χ) reduces a goal $\Gamma \vdash F\psi$ to the following subgoals

- $\Gamma \vdash \chi$
- $\Gamma \vdash G(\chi \wedge \neg\psi \rightarrow X\chi \wedge \text{next}(\varrho) \leq \varrho)$
- $\Gamma \vdash G(\chi \wedge \neg\psi \wedge \sigma \rightarrow \text{next}(\varrho) < \varrho)$
- $\Gamma \vdash G\chi \rightarrow GF\sigma$

In contrast to **LivenessRankInvTac**, the ranking function ϱ will only decrease when the trigger signal σ and the invariant χ holds. The constraint $G\chi \rightarrow GF\sigma$ ensures that every cycle of χ -states must contain a σ state so that $G\chi$ implies that ϱ would contain an infinitely decreasing sequence which contradicts the well-foundedness of ϱ 's domain.

- **LivenessRankSeqTac** [$\varphi_1, \dots, \varphi_n$] This tactic can prove a liveness property by a ranking sequence which satisfies the subgoals:

$$\frac{\Gamma \vdash F\varphi_0 \quad \Gamma \vdash \bigvee_{j=0}^n \varphi_j \quad \bigwedge_{i=0}^{n-1} \Gamma \vdash G(\varphi_{i+1} \rightarrow F\bigvee_{j=0}^i \varphi_j)}{\Gamma \vdash \bigvee_{j=0}^n \varphi_j \quad \bigwedge_{i=0}^{n-1} \Gamma \vdash G(\varphi_{i+1} \rightarrow F\bigvee_{j=0}^i \varphi_j)}$$

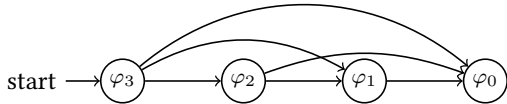
The rule is a repeated application of the following simpler rule (which is clearly correct by noting that $G(\beta \rightarrow F\gamma)$ is equivalent to $G((\beta \vee \gamma) \rightarrow F\gamma)$)

$$\frac{\Gamma \vdash G((\alpha \vee \beta \vee \gamma) \rightarrow F\gamma)}{\Gamma \vdash G(\alpha \rightarrow F(\beta \vee \gamma)) \quad \Gamma \vdash G(\beta \rightarrow F\gamma)}$$

which yields with $\alpha := \varphi_n$, $\beta := \bigvee_{j=1}^{n-1} \varphi_j$, $\gamma := \varphi_0$:

$$\frac{\Gamma \vdash G((\bigvee_{j=0}^n \varphi_j) \rightarrow F\varphi_0)}{\Gamma \vdash \bigwedge_{i=0}^{n-1} G(\varphi_{i+1} \rightarrow F\bigvee_{j=0}^i \varphi_j)}$$

Note that we may not have $G(\varphi_{i+1} \rightarrow F\varphi_i)$ as seen by the example below, but we do have $G(\varphi_i \rightarrow F\varphi_0)$:



In many cases, F can be replaced by X in the subgoals (see **LivenessNextTac**). However, the use of F allows one to have terminating loops inside of one state set φ_i .

- **LivenessWaitTac**(σ, χ)

$$\frac{\Gamma \vdash F\psi}{\Gamma \vdash F\sigma \quad \Gamma \vdash \chi \quad \Gamma \vdash G(\chi \rightarrow (\sigma \rightarrow X\psi) \wedge (\neg\sigma \rightarrow X\chi))}$$

- **LTLTac** proves a goal by an abstraction to a propositional temporal logic (LTL) formula which is translated to a nondeterministic ω -automaton whose emptiness is checked by symbolic model checking using BDDs

- **MpTac**(ψ)

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi \quad \Gamma \vdash \psi \rightarrow \varphi}$$

- **NextTac** drive an outermost temporal next operator inwards to the atoms of its subformulas to prepare an expanding with the delayed assignments

- **NoTac** does nothing, but is useful for tactical like **OrelseTac**.

- **ObligationTac**(χ): To understand this rule, note that $G\varphi \vee F\psi$ is equivalent to $[\varphi \cup F\psi]$ so that we can use $\chi := F\psi$ to prove its correctness and completeness:

$$\frac{\Gamma \vdash G\varphi \vee F\psi}{\Gamma \vdash [\varphi \cup \chi] \quad \Gamma \vdash G(\chi \rightarrow F\psi)}$$

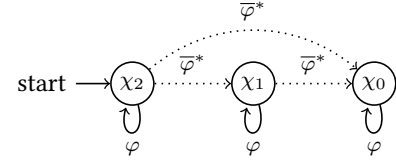
- **PersistenceTac**(χ): Note that the rule below is correct even though $AFG\varphi$ is not equivalent to $AFAG\varphi$.

$$\frac{\Gamma \vdash FG\varphi}{\Gamma \vdash F\chi \quad \Gamma \vdash G(\chi \rightarrow \varphi \wedge X\chi)}$$

- **PersRankSeqTac**($[\chi_0, \dots, \chi_m]$) proves persistence properties by a sequence of predicates which denote unions of SCCs where φ always holds, and which may have finite paths to other such SCCs. It reduces a goal $\Gamma \vdash FG\varphi$ to the following subgoals

- $\Gamma \vdash F\bigvee_{j=0}^m \chi_j$
- $\bigwedge_{i=1}^m \Gamma \vdash G(\chi_i \rightarrow (G\chi_i \vee F\bigvee_{j=0}^{i-1} \chi_j))$
- $\Gamma \vdash G(\chi_0 \rightarrow G\chi_0)$
- $\Gamma \vdash G((\bigvee_{j=0}^m \chi_j) \rightarrow \varphi)$

The situation is illustrated as shown below:



Hint: To prove the generated goals $\Gamma, \chi_i \vdash G\chi_i \vee F\psi$, consider the inputs σ which enforce leaving the SCCs χ_i so that **ObligationTac** with $\chi := \chi_i \wedge \sigma$ generates subgoals $\Gamma, \chi_i \vdash [\chi_i \cup \chi_i \wedge \sigma]$ and $\Gamma, \chi_i, \sigma \vdash F\psi$ which can be proved with **InductWeakUntilTac** and **LivenessBoundTac**, respectively.

- **StrongBeforeTac**

$$\frac{\Gamma \vdash [\varphi \underline{B} \psi]}{\Gamma \vdash [\varphi \underline{B} \psi] \quad \Gamma \vdash F\varphi}$$

- **StrongUntilTac**

$$\frac{\Gamma \vdash [\varphi \underline{U} \psi]}{\Gamma \vdash [\varphi \underline{U} \psi] \quad \Gamma \vdash F\psi}$$

- **StrongWhenTac**

$$\frac{\Gamma \vdash [\varphi \underline{W} \psi]}{\Gamma \vdash [\varphi \underline{W} \psi] \quad \Gamma \vdash F\psi}$$

- **StrongAsWeakTac** applies one of **StrongBeforeTac**, **StrongUntilTac**, or **StrongWhenTac**, depending on the top-level operator of the goal's conclusion
- **SmtZ3Tac**(thmL) instantiates the listed theorems as new assumptions and tries to prove the obtained proof goal using Microsoft's SMT solver Z3. It either proves the goal or generates a counterexample.
- **TautTac**(thmL) instantiates the listed theorems as new assumptions and tries to prove the obtained proof

goal using Microsoft's SMT solver Z3. In contrast to `SmtZ3Tac`, atoms are abstracted as new propositional variables, so that only propositional SAT solving is used.

- `TransImpTac`(α)

$$\frac{\Gamma \vdash \psi \rightarrow \varphi}{\Gamma \vdash \psi \rightarrow \alpha \quad \Gamma \vdash \alpha \rightarrow \varphi}$$

- `TransIffTac`(α)

$$\frac{\Gamma \vdash \psi \leftrightarrow \varphi}{\Gamma \vdash \psi \leftrightarrow \alpha \quad \Gamma \vdash \alpha \leftrightarrow \varphi}$$

- `TransWhenTac`(χ)

$$\frac{\Gamma \vdash G(\alpha \rightarrow [[\beta \text{ W } \gamma] \text{ W } \delta])}{\Gamma \vdash G(\alpha \rightarrow [\chi \text{ W } \delta]) \quad \Gamma \vdash G(\chi \wedge \delta \rightarrow [\beta \text{ W } \gamma])}$$

- `TransEventualTac`(χ)

$$\frac{\Gamma \vdash G(\varphi \rightarrow F\psi)}{\Gamma \vdash G(\varphi \rightarrow F\chi) \quad \Gamma \vdash G(\chi \rightarrow F\psi)}$$

- `UnrollTempOpsTac` unrolls temporal logic operators such that they all will occur under `Next`, `PastWeakNext`, or `PastStrongNext`.

- `UndischAllTac`

$$\frac{\gamma_0, \dots, \gamma_{n-1} \vdash \varphi}{\vdash \gamma_0 \wedge \dots \wedge \gamma_{n-1} \rightarrow \varphi}$$

- `UndischTac`(i)

$$\frac{\gamma_0, \dots, \gamma_{n-1} \vdash \varphi}{\gamma_0, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_{n-1} \vdash \gamma_i \rightarrow \varphi}$$

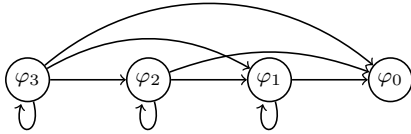
- `UndoTac` undo the previous tactic
- `WaitForTac`[$\chi_0, \dots, \chi_m$]: If the assertion list is empty, the list [$\varphi_0, \dots, \varphi_m$] is used instead:

$$\frac{\Gamma \vdash [\varphi_m \text{ U } \dots [\varphi_1 \text{ U } \varphi_0] \dots]}{\Gamma \vdash \bigvee_{i=0}^m \chi_i}$$

$$\Gamma \vdash G(\bigwedge_{i=0}^m (\chi_i \rightarrow \varphi_i))$$

$$\bigwedge_{i=1}^m \Gamma \vdash G(\chi_i \rightarrow X(\bigvee_{j=0}^i \chi_j))$$

This tactic can be used to prove that a sequence of properties follow one after the other in the specified order where it is allowed that some of them are skipped and that one holds for the rest of the time (skipping its successor properties):



For the correctness, note that $G(\chi_i \rightarrow X(\Upsilon \vee \chi_i))$ implies $G(\chi_i \rightarrow [\chi_i \text{ U } \Upsilon])$: If χ_i holds, then we must have $\Upsilon \vee \chi_i$ at the next point of time. If Υ holds next, we have $[\chi_i \text{ U } \Upsilon]$, and if χ_i holds next, the same argument applies to the next point of time. Even if this continues forever, we still have $[\chi_i \text{ U } \Upsilon]$. Due to $\bigvee_{i=0}^m \chi_i$, at least one of the χ_i must hold, and therefore we also have $[\chi_m \text{ U } \dots [\chi_1 \text{ U } \chi_0] \dots]$. Since `U` is monotonic, the

monotonicity constraints $\bigvee_{i=0}^m G(\chi_i \rightarrow \varphi_i)$ imply that we therefore also have $[\varphi_m \text{ U } \dots [\varphi_1 \text{ U } \varphi_0] \dots]$. The rule is an equivalence if the φ_i exclude each other.

Proof Tacticals

Proof tacticals are 'higher-order tactics', i.e., functions mapping tactics to tactics. Tactics have type `unit -> unit` and their application to `()` adds subgoals to the current goal in the proof tree as a side effect.

- `OrelseTac` `tac1` `tac2` applies the given tactic `tac1` and if that fails, it applies `tac2`.
- `RepeatTac` `tac` applies the given tactic `tac` repeatedly until the application no longer changes the goal.
- `ThenL` `tac` `tacL` applies the given tactic `tac` to generate subgoals `g[0..n-1]` and then applies tactic `tacL[i]` to `g[i]`.
- `ThenAll` `tac1` `tac2` applies tactic `tac1` to the current subgoal and applies then `tac2` to all of the subgoals generated by `tac1`