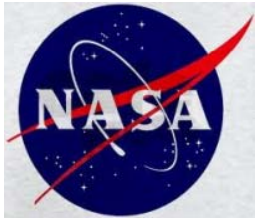
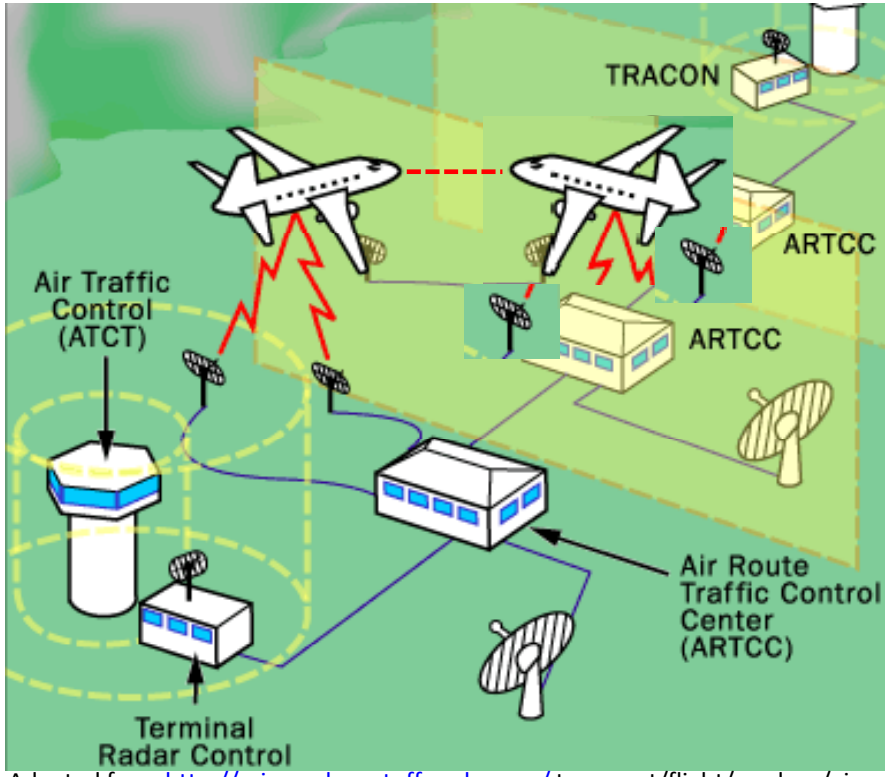
Integrated Model Checking and Simulation of NextGen Authority and Autonomy (NextGenAA)

Ellen J. Bass, Karen Feigh, Elsa Gunter, John Rushby
Matthew Bolton, Dennis Griffith, William Mansky

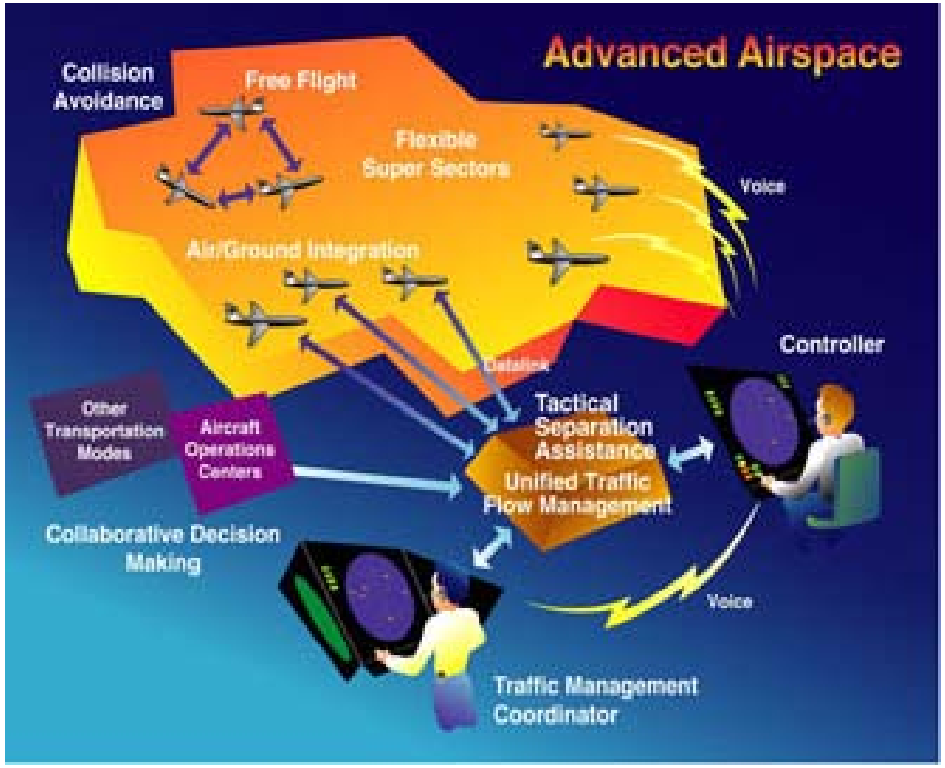


NNA10DE79C

NextGen systems are envisioned to be composed of human and automated agents interacting with dynamic flexibility in the allocation of authority and autonomy.



Adapted from http://science.howstuffworks.com/ transport/flight/modern/air-traffic-control2.htm

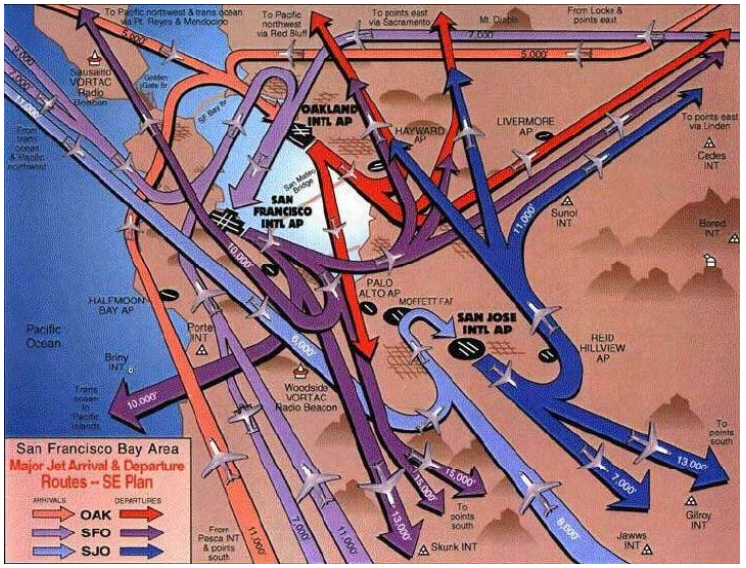


http://www.aviationsystemsdivision.arc.nasa.gov/research/tactical/index.shtml

Analysis of concepts of operation requires methods for verifying and validating **range of roles and responsibilities** potentially assignable to **human and automated agents**

- Automation
- Human behavior and operational procedures
- Methods of collaboration
- Organization structures
- Policies and regulations
- Environment
- …

http://www.boeing.com/news/frontiers/archive/2006/april/i_ca3.html



http://static.howstuffworks.com/gif/air-traffic-control-baytracon-east2.jpg

Agent-based simulation models such complexity

- Tradeoff between fidelity and the number of simulation runs

Model checking techniques can verify safety properties

- Component models are of sufficiently limited scope
- By analyzing simulation traces, model checking ensure simulation's design meets the intended analysis goals

Need:

Common representations for model components

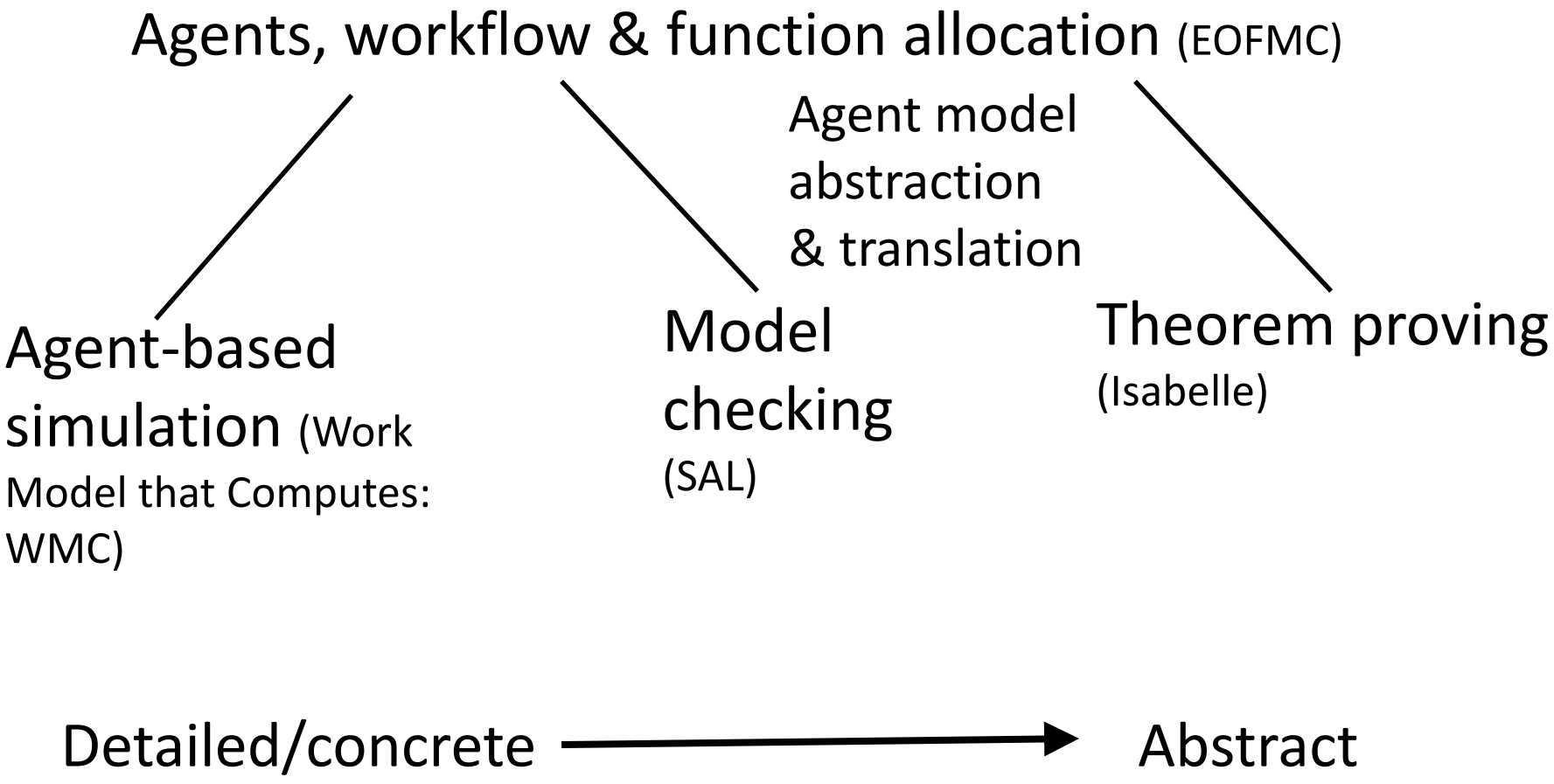
Techniques for determining the set of analyses to run

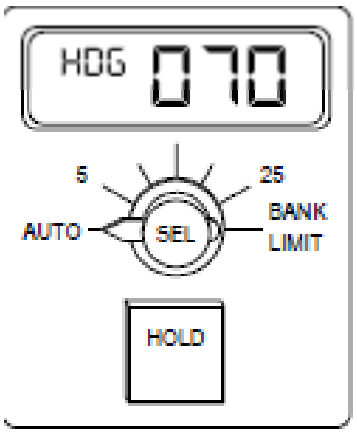
Long term objectives

- **Unified agent modeling language**
  - -Used by agent based simulation and model checking for autonomy and authority
- **Approach that allows formal model-checking techniques to verify the bounds of human behavior and accountability in a distributed human-automation systems**
- **Integration of agent based simulation and formal methods**
  - -Identify system level problems while providing insight on how to correct the models used in each
- **Integration of formal model-checking with agent-based languages for modeling and simulating autonomy and authority via model-based generation of scenarios**

Current Approach

Link theorem proving, model checking and simulation through top level XML description of agents, workflow and function allocation





ATCo PF PM PF & PM All 3

precondition IATCoSelectedClearance /= NULL

aChange Heading

ord

decomposition operator

IATCoSelectedClearance /= IATCoHeadingHeardFromPilots

aCommunicate AndConfirm Heading

IATCoSelectedClearance = IATCoHeadingHeardFromPilots

aSetNew Heading

ord

aToggle ATCoTalk | aCom Heading | aToggle ATCoTalk | aToggle PMTalk | aComConfirm Heading | aToggle PMTalk

ord / com / ord / ord / com / ord

hARCoPressOrReleaseSwitchToTalk

hATCoTalk IATCoSelectedClearance | iPFHeadingFromATC | iPMHeadingFromATC

hARCoPressOrReleaseSwitchToTalk

hPMPressOrReleaseSwitchToTalk

hPMTalk IPMHeadingFromATC | IATCoHeadingHeardFromPilots | iPFHeadingFromPM

hPMPressOrReleaseSwitchToTalk

ord

repeatcondition IPFHeadingFromPM /= IPFHeadingFromATCo

aChangeAndConfirm

IPFHeadingFromPM = IPFHeadingFromATCo **completioncondition**

aExecuteTheChange

ord / ord

aMakeTheChange | aConfirmTheChange | hPFEngageHeadingSelection

ord / and_par

hPFPushHeadingSelectKnob | hPFRotateToHeadingcSayableToHeading (IPFHeadingFromATC) | hPFPullHeadingSelectKnob

aPointAtHeadingWindow | aSayTheHeading

com / com

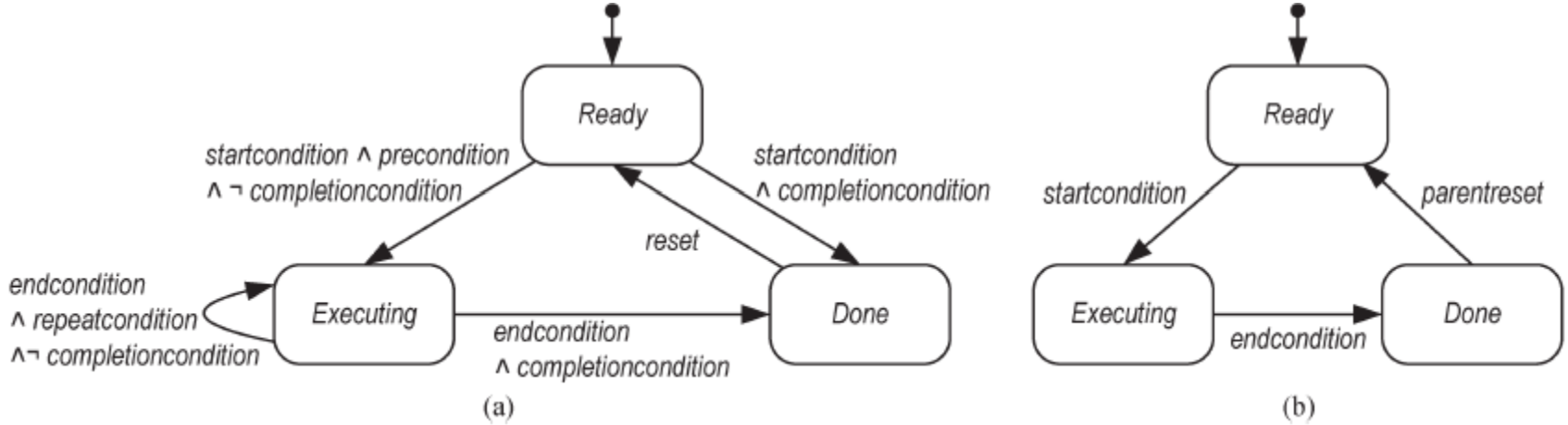
hPMPointHeadingWindow | iPFThingPointedAtByPM

hPMTalk cHeadingToSayable (iHeadingWindowHeading) | IPFHeadingFromPM

Decomposition operators

Operator Type	Semantics	Modality	
		Sequential	Parallel
and	All of the sub-acts must execute	*and_seq*	*and_par*
or	One or more of the sub-acts must execute	*or_seq*	*or_par*
optor	Zero or more of the sub-acts must execute	*optor_seq*	*optor_par*
xor	Exactly one sub-act must execute	*xor*	—
ord	All sub-acts must execute in the order they appear in	*ord*	—
sync	All sub-acts must be executed at the same time	—	*sync*

Translation to SAL



(a) Execution state transition diagram for a generic activity. (b) Execution state transition diagram for a generic action.

Implicit transitions based on position in the task structure, and execution state of its parent, sub-acts, and siblings:

startcondition: triggers the start of an activity or action defined in terms of the execution states of its parent and siblings.

endcondition: ends the execution of an activity or action defined in terms of the execution state of its sub-acts.

reset: resets an activity (have it return to the Ready execution state).

Transition Rules for Theorem Proving

$$\frac{\mathsf{start}(t[]) \qquad \mathsf{val}_{lenv,\mathsf{hd}(envs)}(v) = w}{sys, t[] \vdash (envs, lenv, (\mathsf{HumanAction}(a, v), \mathsf{Ready})) \xrightarrow{\{(a,w)\}} (\mathsf{hd}(envs) + \{a \mapsto w\}@envs, lenv, (\mathsf{HumanAction}(a, v), \mathsf{Executing}))}$$

Operator performs operation, causing output variable value to change from *a* to *w*, and the action to transition to the *Executing* state

$$\frac{\mathsf{start}(t[]) \qquad \mathsf{val}_{lenv,\mathsf{hd}(envs)}(v) = w}{sys, t[] \vdash (envs, lenv, (\mathsf{LocVar}(x, v), \mathsf{Ready})) \longrightarrow (envs, lenv + \{x \mapsto w\}, (\mathsf{LocVar}(x, v), \mathsf{Executing}))}$$

Operator associates value *v* with variable *x* (remembering or making a note of a value)

$$\frac{\mathsf{start}(t[]) \qquad \mathsf{val}_{lenv,\mathsf{hd}(envs)}(v) = w}{sys, t[] \vdash (envs, lenv, (\mathsf{Com}(n, v, x_1, \ldots, x_m), \mathsf{Ready})) \longrightarrow (envs, lenv + \{x_1, \ldots, x_m \mapsto w\} (\mathsf{Com}(n, v, x_1, \ldots, x_m), \mathsf{Executing}))}$$

Each of the local variables (belonging to various humans) are updated with the value of *v*

$$\frac{}{sys, t[] \vdash (envs, lenv, (action, \mathsf{Executing})) \longrightarrow (envs, lenv, (action, \mathsf{Done}))}$$

Any action that is *Executing* may immediately transition to *Done*

$$\frac{}{sys, t[] \vdash (envs, lenv, (action, state)) \longrightarrow (sys(envs)@envs, lenv, (action, state))}$$

System may at any time act by updating the environment

Agent Modeling

- Simulation framework to simulate work in complex, heterogeneous dynamic systems (humans, physical systems, computer agents and regulations, …)
- Based on WMC (Work Model that Computes), a work model that computes is a computer simulation that includes:
  - All aspects of the work model system (all behaviors represented at all levels of abstraction)
  - Interactions between multiple agents
  - Interactions between agents and environment

WMC Constructs

Agent: entity that performs an action.

Action: work performed by an agent at one instance in time.

Resource: a specific state of the environment.

Environment: collection of resources available for interaction with the agent.

Decision actions: process of selecting a course of action based on the environmental context.

Temporal actions: actions initiated by the agent. It obtains a specific resource from the environment and changes its value.

Functions: describes how something may be achieved (in the coding sense). It can call upon other functions or temporal actions.

Abstraction Hierarchy

Functional purpose: Purposes for which the system was designed as well as the external constraints on its operation.

Abstract functions: The criteria that the work system uses for measuring its progress towards the functional purposes.

General functions: Basic functions that the system is designed to achieve in order to accomplish the higher functional purposes.

Temporal functions: Collections of temporal actions

Resources: Specific instantiations of environmental variables

Formal Modeling and Analysis for Interactive Hybrid Systems

Approach for very **abstract** modeling of **hybrid systems**

Relational approximations and automated analysis using **infinite BMC** supported by an **Satisfiability Modulo Theories** (SMT) solver

Propositional satisfiability (SAT) solving can be generalized to SMT, which supports real numbers, mathematical (i.e., unbounded) integers, and uninterpreted functions.

Formal Modeling and Analysis for Interactive Hybrid Systems

If anomalies are discovered, decide if due to approximations or "real"

-When counterexamples are found, additional constraints direct counterexamples toward plausible scenarios

Use this scenario to guide a limited search in a simulator to see if a similar anomalous scenario can be created in high fidelity

Mental model of automation

- Frequential simplification causes rarely taken transitions, or rarely encountered guards on transitions, to be forgotten
- Inferential simplification causes transition rules that are “similar” to one another to be merged into a single prototypical rule

Based on Javaux 1998

A320 example

Several vertical modes and submodes

V/S FPA (Flight Path Angle submode of Vertical Speed mode)

OP CLB (Climb submode of Open mode)

OP DES (Descent submode of Open mode)

Several autothrottle modes and submodes

SPD (Speed mode)- tries to maintain a specific airspeed set Flight Control Unit (FCU)

Here in steep descent, FPA is prioritized over airspeed

- Can exceed the speed set in the FCU
- If exceeds the maximum safe speed, automated speed protection switches vertical mode
  - V/S FPA -> OP DES or OP CLB, which prioritize airspeed over vertical speed
    - Based on “target altitude” set in the FCU
    - FCU ALT > current altitude -> OP CLB; use max thrust and FPA to maintain speed

Pilots

Nondeterministic choice among

- Extending or retracting the flaps (when the mental model is descend or climb, respectively),
- Dialing a (nondeterministic) value into FCU ALT,
- Switching mental model to descend or climb, or
- Doing nothing

When mental model switches to descend or climb,

FCU mode is set to VS FPA and

Nondeterministic negative or positive flight path angle, respectively, is dialed into FCU FPA

Aircraft

Uninterpreted functions (actually relations) describe the dynamics of the airplane: airspeed and altitude

- These functions input current airplane airspeed, altitude, engine thrust, and pitch angle and output sets (modeled as predicates—i.e., functions with range type BOOLEAN) of airspeed or altitude as appropriate
- 2 modes, depending on whether or not the flaps are extended
- Relational models
  - Altitude must increase when pitch angle > 0 (new value chosen nondeterministically from those values greater than its current value)

Automation

Uninterpreted functions computes engine thrust and pitch angle

Inputs

- Pilot inputs (desired vertical mode, FCU altitude, FCU FPA, flap setting)
- Current aircraft state (airspeed and altitude),

Processing

- Determines actual vertical mode to be used (may differ from that desired by pilots if a protection is being applied)

Outputs

- Applies control laws to determine thrust and pitch settings

Mode = VS FPA and airspeed > Max speed

Next mode:

If FCU altitude > altitude Then is OP CLB Else OP DES

Mode = VS FPA and airspeed <= Max speed

Pitch is governed by VS FPA pitch law

Mode = OP CLB

Pitch is governed by OP CLB pitch law

Constraints

Enforces suitable relations on

- Altitude and airspeed (interpretation of the airplane model)
- Thrust and pitch (interpretation of the automation)
- Automation surprise
- Separate model that observes the state of the system and sets Boolean when it observes a violation of the relational constraint

OP DES and pitch > 0

OP CLB and pitch < 0

VS FPA and FCU FPA <= 0 and pitch > 0

VS FPA and FCU FPA >= 0 and pitch < 0

Pitch > 0 and altitude decreasing

…

Synchronous observer

Separate model that observes the state of the system and sets Boolean when it observes a violation of the relational constraint

Counter Example

Mode “Other”, IAS 200, ALT 3000 < FCU,
Flap ret., **Mental Model Level, Pitch 0**

Flying clean, dial descend, FCU FPA <0, FCU ALT 3200,
Mode “VS FPA”, IAS 201, ALT 2990 < FCU,
Flap ret., **Mental Model Desc., Pitch <0**

Flying clean, VS FPA, Extend flaps, Max Speed 180
Mode “VS FPA”, IAS 200, ALT 2988 < FCU,
Flap ext., **Mental Model Desc., Pitch <0**

Flying with flaps, mode reversion
Mode “OP CLB”, IAS 201, ALT 2988 <FCU,
Flap ext., **Mental Model Desc., Pitch 0**

Flaps, OP CLB
Mode “OP CLB”, IAS 200, ALT 2990 <FCU,
Flap ext., **Mental Model Desc., Pitch>0**

Next steps

- Formal description of data abstractions used in NextGen simulations
- Example protection envelopes for authority and autonomy for all major components
- Transition semantics for core agent-based language
- Initial prototype of analysis toolset

Long term

- Formal analysis of simulation traces
- Model guided simulation
- Integrate agent based simulation, protection envelope trace analyses, and model checking with counterexample guided abstraction refinement into a single coherent process

For more information

Bass, E.J., Feigh, K.,M., Gunter, E. & Rushby, J. (accepted). Formal modeling and analysis for interactive hybrid systems. *4th International Workshop on Formal Methods for Interactive Systems (FMIS),* June 21, 2011, Limerick, Ireland.

Bass, E.J., Bolton, M.L., Feigh, K., Griffith, D., Gunter, E., Mansky, W., & Rushby, J. (under review). Toward a multi-method approach to formalizing human-automation interaction and human-human communications. *2011 IEEE International Conference on Systems, Man, and Cybernetics*. October 9-12, 2011, Anchorage, Alaska.

Bolton, M.L., Siminiceanu, R.I., & Bass, E. J. (in press). A systematic approach to model checking human-automation interaction using task-analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A : Systems and Humans*.

Pritchett A.R. Kim, S.Y., Kannan S. & K.M. Feigh (2011). Simulating situated work. In 2011 IEEE Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA),