Sure, let's create a security case for ACAS Xu (Airborne Collision Avoidance System Xu) using the given security case pattern predicates. We'll fill in the placeholders specific to ACAS Xu and structure the case hierarchically:

```

- G0: ACAS Xu satisfies security requirements

- C0: Description of ACAS Xu

- C1: SR are requirements about protecting the system from malicious entities

- J0: The argumentation is based on satisfaction of SRs

- A0: System SRS are complete, adequate, and consistent

- S0: Argue through asset protection and secure development requirements

- G1: ACAS Xu satisfies the asset protection requirements

- A1: Asset inventory is established

- S1: Argue through the different stages of the system development life cycle

- G3: Asset protection requirements are met during the architecture design phase

- C2: Description of the ACAS Xu architecture

- S2: Argue through deriving security threats from SRs

- G5: ACAS Xu architecture is protected against identified security threats (STs)

- A2: All relevant threats have been identified

- S3: Argue over each security threat

- G0.X: ACAS Xu architecture is protected against STX

- G4: Asset protection requirements are met during other phases (undeveloped)

- G2: ACAS Xu satisfies secure development requirements (undevelopStantiated)

- G6: ACAS Xu architecture is validated (undevelopStantiated)

- C3: Description of ACAS Xu architecture model

- A3: ACAS Xu architecture model is well defined in formal method (uninstantiated)

```

For clarity:

- Each level of hierarchy is denoted with a dash (`-`).

- Placeholders for `ACAS Xu` have been filled in.

- Undeveloped and uninstantiated elements are marked accordingly.

This hierarchical representation helps visualize the structure of the security case for ACAS Xu, with each element's dependencies and contextual elements clearly laid out.