High-level structure of the code

```
covers(P,C) :-
  ~exclusion_applies(C) &
  ~no_leave_extension_filed(C) &
  ~coverage_ceased(C) &
  eligible_person(C) & ~policy_not_in_effect(C) & ~claim_date_invalid(C)
```

3 (Who can have insurance under this policy?)

3.1 (Eligibility for cover)

Metadata

```
% Type of person
attribute(tal_death_benefit, claim.person_type)
codomain(claim.person_type, person_type)
changestyle(claim.person_type, selector)

attribute(tal_death_benefit, claim.person_au_citizen)
codomain(claim.person_au_citizen, boolean)
changestyle(claim.person_au_citizen, selector)

attribute(tal_death_benefit, claim.person_nz_citizen)
codomain(claim.person_nz_citizen, boolean)
changestyle(claim.person_nz_citizen, selector)

attribute(tal_death_benefit, claim.person_au_living_working)
codomain(claim.person_au_living_working, boolean)
changestyle(claim.person_au_living_working, selector)

attribute(tal_death_benefit, claim.person_au_perma_resident)
codomain(claim.person_au_perma_resident, boolean)
changestyle(claim.person_au_perma_resident, selector)

attribute(tal_death_benefit, claim.person_visa_gain_employ)
codomain(claim.person_visa_gain_employ, boolean)
changestyle(claim.person_visa_gain_employ, selector)

attribute(tal_death_benefit, claim.person_other_eligible_requirements)
codomain(claim.person_other_eligible_requirements, boolean)
changestyle(claim.person_other_eligible_requirements, selector)

% claim.claimant_age and claim.benefit_ceasing_age are defined in 14.1
```

Library

```
covers(P,C) :-
```

eligible_person(C)

eligible_person(C) :-
person_type(C) & person_au_resident(C,yes) & person_within_age_range(C,yes) &
claim.person_other_eligible_requirements(C,yes)
% should claim.person_other_eligible_requirements be in a negative form so as to avoid user confusion?

person_type(C):-
claim.person_type(C,employee)
person_type(C):-
claim.person_type(C, contractor)
person_type(C):-
claim.person_type(C, member)
person_type(C):-
claim.person_type(C, spouse)

person_au_resident(C) :-
claim.person_au_citizen(C,yes)
person_au_resident(C) :-
claim.person_nz_citizen(C,yes) & claim.person_au_living_working(C,yes)
person_au_resident(C) :-
claim.person_au_perma_resident(C,yes)
%Ignored an appropriate Visa element, as people usually have an appropriate Visa
person_au_resident(C) :-
claim.person_au_living_working(C,yes) & claim.person_visa_gain_employ(C,yes)

person_within_age_range(C) :-
claim.claimant_age(Y) & claim.benefit_ceasing_age(C,A) & leq(Y,A) & leq(16,Y)

World
type(employee, person_type)
type(contractor, person_type)
type(member, person_type)
type(spouse, person_type)
type(others, person_type)
%I assume that if a user chooses “others”, person_type will be false.

9 (Employer approved leave)

9.1 (Cover whilst on employer approved leave)

Metadata
attribute(tal_death_benefit, claim.employer_approved_leave)
codomain(claim.employer_approved_leave, boolean)
changestyle(claim.employer_approved_leave, selector)

attribute(tal_death_benefit, claim.employer_approved_leave_elapsed)
changestyle(claim.employer_approved_leave_elapsed, numberstyle)

attribute(tal_death_benefit, claim.extend_cover_app_prior_to_expiry)
codomain(claim.extend_cover_app_prior_to_expiry, boolean)
changestyle(claim.extend_cover_app_prior_to_expiry, selector)

Library
covers(P,C) :-
~no_leave_extension_filed(C)

no_leave_extension_filed(C) :-
claim.employer_approved_leave(C, yes) & claim.employer_approved_leave_elapsed(C, M) & ~leq(M, 24) & claim.extend_cover_app_prior_to_expiry(C, no)

Explanation
We encoded that coverage ceases if employees are on approved leave for longer than 24 months without an application of extended cover prior to expiry. Since unpaid premiums leading to the policy ceasing were already encoded in section 11.1, we did not encode that here. The employee not returning is not encoded here as it is encoded in section 14. We did not encode continued payments because we have already encoded that unpaid premiums lead to the policy not applying.

10 (Overseas cover)

10.1 (Overseas cover)

We did not encode this as it simply states that cover applies internationally, apart from cases in which the claimant does not meet the eligibility criteria or has unpaid premiums, neither of which is specific to international coverage, both of which we have encoded elsewhere.

10.2 (Underwriting overseas)

We did not encode this as there is nothing relating to coverage not applying. It simply states that one need not return to Australia to apply for underwritten cover.

10.3 (Assessment of a claim overseas)

We did not encode this as it only dictates that an overseas insuree must provide medical evidence for claim adjudication and that the insurer may require them to return to Australia for adjudication. Neither of these statements has to do with the *conditions that need to be met* for a favorable adjudication.

11 (Exclusions, restrictions and limitations)

11.1 (Self-harm act or injury exclusion)

Metadata

```
attribute(tal_death_benefit, claim.self_inflicted)
codomain(claim.self_inflicted, boolean)
changestyle(claim.self_inflicted, selector)

attribute(tal_death_benefit, claim.months_under_acc)
changestyle(claim.months_under_acc, numberstyle)

attribute(tal_death_benefit, claim.months_under_rein)
changestyle(claim.months_under_rein, numberstyle)

attribute(tal_death_benefit, claim.under_inc)
codomain(claim.under_inc, boolean)
changestyle(claim.under_inc, selector)

attribute(tal_death_benefit, claim.months_last_under_inc)
changestyle(claim.months_last_under_inc, numberstyle)
```

Library

```
exclusion_applies(C) :-
  claim.self_inflicted(C,yes) &
  self_harm_window(C)

self_harm_window(C) :-
  claim.months_under_acc(C,M1) &
  leq(M1,13)
self_harm_window(C) :-
  claim.months_under_rein(C,M2) &
  leq(M2,13)
self_harm_window(C) :-
  claim.under_inc(C,yes) &
  claim.months_last_under_inc(C,M3) &
  leq(M3,13)
```

Explanation

We assessed whether or not the self-inflicted act or injury exclusion applies. It applies exactly when the claimant inflicted self-harm upon themselves (encoded by the attribute claim.self_inflicted) AND if that harm was inflicted within a period of 13 months (encoded by self_harm_window) of the date of acceptance of the Underwritten Cover (claim.months_under_acc) OR the date the Underwritten Cover was reinstated (claim.months_under_rein) OR the date the last date the Underwritten Cover increased (claim.months_last_under_inc) (IF the Underwritten Cover has increased, encoded by claim.under_inc).

11.2 dictates which type of cover applies which we will encode seperately

11.3 dictates which type of cover applies which we will encode seperately

11.4

Explanation

We did not encode this as it relates to recalculations of payment amounts and not whether the death benefit applies.

11.5

Explanation

We did not encode this as it relates to payment amounts, specifically payment amounts dependent on the type of coverage, and not whether the death benefit applies.

11.6

Explanation

We did not encode this as it relates to recalculations of payment amounts and not whether the death benefit applies.

11.7

Explanation

We did not encode this as it relates to recalculations of payment amounts and not whether the death benefit applies.

11.8

Explanation

We did not encode this as it relates to recalculations of payment amounts and not whether the death benefit applies.

11.9

Explanation

did not encode maximum cover limit because it relates to how much TAL will pay, and not whether the death benefit is covered.

11.10

Metadata

attribute(tal_death_benefit, claim.misrepresentation)
codomain(claim.misrepresentation, boolean)
changestyle(claim.misrepresentation, selector)

Library

exclusion_applies(C) :- claim.misrepresentation(C,yes)

Explanation

If there is a misrepresentation TAL does not have to pay the benefit. Misrepresentation is broken down into false answers, partial truths, and answers which don't fairly reflect the truth. However, this is too complex to ask a user not versed in law to subjectively answer.

11.11

Metadata

attribute(tal_death_benefit, claim.unpaid_premiums)
codomain(claim.unpaid_premiums, boolean)
changestyle(claim.unpaid_premiums, selector)

Library

exclusion_applies(C) :- claim.unpaid_premiums(C,yes)

Explanation

If there are unpaid premiums, TAL reserves the right to indefinitely delay payment (in which case it can be considered that the payment does not apply)

11.12

Not encoding 11.12 because the implication of TAL overpaying or not reducing a benefit is that they can "reduce" future payments. Since our encoding is focused on whether the benefits apply and not their amount, full or reduced, this section is irrelevant to our objective.

14 (When Cover Ceases?)

14.1 (Cessation of Cover)

Metadata

% Policy termination date passed
attribute(tal_death_benefit, claim.policy_termin_date_passed)
codomain(claim.policy_termin_date_passed, boolean)
changestyle(claim.policy_termin_date_passed, selector)

% Older than benefit ceasing age
attribute(tal_death_benefit, claim.benefit_ceasing_age)
codomain(claim.benefit_ceasing_age, benefit_ceasing_age)
changestyle(claim.benefit_ceasing_age, selector)

attribute(tal_death_benefit, claim.claimant_age)
changestyle(claim.claimant_age, numberstyle)

% Cease to meet requirements
attribute(tal_death_benefit, claim.req)
codomain(claim.req, boolean)
changestyle(claim.req, selector)

% Cease to meet requirements if the insured is a spouse

attribute(tal_death_benefit, claim.req_spouse)
codomain(claim.req_spouse, boolean)
changestyle(claim.req_spouse, selector)

% Extended cover ceased
attribute(tal_death_benefit, claim.extend_cover_cease)
codomain(claim.extend_cover_cease, boolean)
changestyle(claim.extend_cover_cease, selector)

% Terminal illness benefit already paid
attribute(tal_death_benefit, claim.term_ill_benefit_paid)
codomain(claim.term_ill_benefit_paid, boolean)
changestyle(claim.term_ill_benefit_paid, selector)

% TPD benefit already paid
attribute(tal_death_benefit, claim.total_perm_disable_paid)
codomain(claim.total_perm_disable_paid, boolean)
changestyle(claim.total_perm_disable_paid, selector)

% Death benefit already paid
attribute(tal_death_benefit, claim.death_been_paid)
codomain(claim.death_been_paid, boolean)
changestyle(claim.death_been_paid, selector)

% Claim canceled by the policy owner
attribute(tal_death_benefit, claim.cancel)
codomain(claim.cancel, boolean)
changestyle(claim.cancel, selector)

% Eligibility criteria no longer met
attribute(tal_death_benefit, claim.criteria_not_met)
codomain(claim.criteria_not_met, boolean)
changestyle(claim.criteria_not_met, selector)

% Service in armed forces commenced
attribute(tal_death_benefit, claim.armed_forces)
codomain(claim.armed_forces, boolean)
changestyle(claim.armed_forces, selector)

% Individual life insurance policy already issued
attribute(tal_death_benefit, claim.indiv_life_insur)
codomain(claim.indiv_life_insur, boolean)
changestyle(claim.indiv_life_insur, selector)

% Employer approved leave exceeded does not need to be encoded since it already has been in section 9

Library

```
covers(P,C) :-
~coverage_ceased(C)

% Policy termination date passed
coverage_ceased(C) :-
claim.policy_termin_date_passed(C, yes)
% Older than benefit ceasing age
ceasing_age(C, A) :- claim.benefit_ceasing_age(C, A)
coverage_ceased(C) :-
ceasing_age(C, A) & claim.claimant_age(Y) & ~leq(Y, A)


coverage_ceased(C) :-
claim.req(C, yes)
coverage_ceased(C) :-
claim.req_spouse(C, yes)
coverage_ceased(C) :-
claim.extend_cover_cease(C, yes)
coverage_ceased(C) :-
claim.term_ill_benefit_paid(C, yes)
coverage_ceased(C) :-
claim.total_perm_disable_paid(C, yes)
coverage_ceased(C) :-
claim.death_been_paid(C, yes)
coverage_ceased(C) :-
claim.cancel(C, yes)
coverage_ceased(C) :-
claim.criteria_not_met(C, yes)
coverage_ceased(C) :-
claim.armed_forces(C, yes)
coverage_ceased(C) :-
claim.indiv_life_insur(C, yes)
% No need to encode exceeded leave since this is encoded in section 9
```

World

```
% Benefit ceasing age
type(65, benefit_ceasing_age)
type(70, benefit_ceasing_age)
```

Explanation
There are many dates past which coverage will have ceased, each of which are covered in this section of the encoding. In particular, we encode a dropdown menu of the two possible benefit ceasing ages for the death policy, 65 and 70, and check that if the age of the claimant is greater than this, the policy does not apply. Other exclusions in this section include that a terminal illness or TPD benefit has already been paid.

15 (When the Policy Ends)

15.1 (Duration of the Policy)

Metadata

```
% Claim date, premium due date, policy termination date, and whether overdue has been paid
attribute(tal_death_benefit, claim.claim_date)
changestyle(claim.claim_date, datestyle)
attribute(tal_death_benefit, claim.premium_due_date)
changestyle(claim.premium_due_date, datestyle)
attribute(tal_death_benefit, claim.overdue_amount_paid)
codomain(claim.overdue_amount_paid, boolean)
changestyle(claim.overdue_amount_paid, selector)
attribute(tal_death_benefit, claim.policy_termination_date)
changestyle(claim.policy_termination_date, datestyle)

% Has the owner terminated the policy
attribute(tal_death_benefit, claim.owner_teriminated)
codomain(claim.owner_teriminated, boolean)
changestyle(claim.owner_teriminated, selector)

%Has the insurer terminated the policy
attribute(tal_death_benefit, claim.insurer_terminated)
codomain(claim.insurer_terminated, boolean)
changestyle(claim.insurer_terminated, selector)

%How many days since the insurer provided notice
attribute(tal_death_benefit, claim.days_since_notice_provided)
changestyle(claim.days_since_notice_provided, numberstyle)

%Has the last benefit been paid
attribute(tal_death_benefit, claim.last_benefit_paid)
codomain(claim.last_benefit_paid, boolean)
changestyle(claim.last_benefit_paid, selector)

%Has cover ended
attribute(tal_death_benefit, claim.cover_ended)
codomain(claim.cover_ended, boolean)
changestyle(claim.cover_ended, selector)
```

Library

```
covers(P,C) :-
~claim_date_invalid(C)

% Checks to see if claim date is between the premium due date and policy expiration date
claim_date_invalid(C) :-
claim.overdue_amount_paid(C, no) &
claim.claim_date(C, D1) &
claim.premium_due_date(C, D2) &
claim.policy_termination_date(C, D3) &
symleq(D1, D3) &
symleq(D2, D1)

covers(P,C) :-
~policy_not_in_effect(C)

%Has owner terminated the policy
policy_not_in_effect(C) :-
claim.owner_teriminated(C, yes)

%Checks to see if insurer terminated and at least 30 days notice has elapsed
policy_not_in_effect(C) :-
claim.insurer_terminated(C, yes) & claim.days_since_notice_provided(C, M) & leq(30, M)

%Checks to see if the user is finished with the insurance and has made their final payment
policy_not_in_effect(C) :-
claim.last_benefit_paid(C, yes)

%Checks to see if coverage has ended
policy_not_in_effect(C) :-
claim.cover_ended(C, yes)
```

Explanation

Encoded that claims filed after the premium due date and before the policy termination will not be covered unless the overdue premium is paid. Aslo encoded the various reasons the policy duration can end, such as the user terminating the contract, coverage ending, and the insurer terminating the contract provided they have given at least 30 days of notice to the user.

15.2 (Cash Value on Termination)

Explanation

Didn’t encode as it only relates to cash value and not whether death benefit applies.