Stable Matching

Peter Gammie

March 19, 2025

Abstract

We mechanize proofs of several results from the *matching with contracts* literature, which generalize those of the classical two-sided matching scenarios that go by the name of *stable marriage*. Our focus is on game theoretic issues. Along the way we develop executable algorithms for computing optimal stable matches.

Contents

1 Introduction

As economists have turned their attention to the design of such markets as school enrolments, internships, and housing refugees (Andersson and Ehlers 2016), particular *matching* scenarios have proven to be useful models. Roth (2015) defines matching as "economist-speak for how we get the many things we choose in life that also must choose us," and one such two-sided market is now colloquially known as the *stable marriage problem*. It was initially investigated by Gale and Shapley (1962), who introduced the key solution concept of *stability*, and the *deferred-acceptance algorithm* that efficiently constructs stable matches for it. We refer readers unfamiliar with this classical work to §2, where we formalize this scenario and mechanize a non-constructive existence proof of stable matches due to Sotomayor (1996). Further in-depth treatment can be found in the very readable monographs by Gusfield and Irving (1989) (algorithmics), Roth and Sotomayor (1990) (economics), and Manlove (2013).

Recently Hatfield and Milgrom (2005) (see also Fleiner (2000, 2002, 2003)) have recast the two-sided matching model to incorporate *contracts*, which intuitively allow agents to additionally indicate preferences over conditions such as salary. By allowing many-to-one matches, some aspects of a labour market can be modelled. Their analysis leans heavily on the lattice structure of the stable matches, and yields pleasingly simple and general algorithms (§5). Later work trades this structure for generality, and the analysis becomes more intricate (§6). The key game-theoretic result is the (one-sided) strategy-proofness of the optimal stable match (§8).

This work was motivated by the difficulty of navigating the literature on *matching with contracts* by non-specialists, as observed by Caminati et al. (2015a,b). We impose some order by formalizing much of it in Isabelle/HOL (Nipkow et al. 2002), a proof assistant for a simply-typed higher-order logic. By carefully writing definitions that are executable and testable, we avail ourselves of Isabelle's automatic tools, specifically `nitpick` and `sledgehammer`, to rapidly identify errors when formulating assertions. We focus primarily on strategic (game theoretic) issues, but our development is also intended to serve as a foundation for further results.

The proof assistant forces us to take care of all details, which yields a verbosity that may deter some readers. We suggest that most will fare best by reading the definitions and **lemma/theorem** statements closely, and skipping the proofs. (The important results are labelled **theorem** and **proposition**, but often the **lemma**s contain the meat.) The material in §4 on choice functions is mostly for reference.

This PDF is generated directly from the development's sources and is extensively hyperlinked, but for some purposes there is no substitute to firing up Isabelle.

2 Sotomayor (1996): A non-constructive proof of the existence of stable marriages

We set the scene with a non-constructive proof of the existence of stable matches due to Sotomayor (1996). This approach is pleasantly agnostic about the strictness of preferences, and moreover avoids getting bogged down in reasoning about programs; most existing proofs involve such but omit formal treatments of the requisite assertions. This tradition started with Gale and Shapley (1962); see Bijlsma (1991) for a rigorous treatment.

The following contains the full details of an Isabelle/HOL formalization of her proof, and aims to introduce the machinery we will make heavy use of later. Further developments will elide many of the more tedious technicalities that we include here.

The scenario consists of disjoint finite sets of men M and women W, represented as types $'m{::}\mathit{finite}$ and $'w{::}\mathit{finite}$ respectively. We diverge from Sotomayor by having each man and woman rank only acceptable partners in a way that is transitive and complete. (Here completeness requires *Refl* in addition to *Total* as the latter does not imply the former, and so we end up with a total preorder.) Such orders therefore include cycles of indifference, i.e., are not antisymmetric.

Also matches are treated as relations rather than functions.

We model this scenario in a **locale**, a sectioning mechanism for stating a series of lemmas relative to a set of fixed variables (**fixes**) and assumptions (**assumes**) that can later be instantiated and discharged.

type-synonym $('m, 'w)\ \mathit{match} = ('m \times 'w)\ \mathit{set}$

locale *StableMarriage* =
  fixes $\mathit{Pm} :: 'm{::}\mathit{finite} \Rightarrow 'w{::}\mathit{finite}\ \mathit{rel}$
  fixes $\mathit{Pw} :: 'w \Rightarrow 'm\ \mathit{rel}$
  assumes *Pm-pref*: $\forall m.\ \mathit{Preorder}\ (\mathit{Pm}\ m) \wedge \mathit{Total}\ (\mathit{Pm}\ m)$
  assumes *Pw-pref*: $\forall w.\ \mathit{Preorder}\ (\mathit{Pw}\ w) \wedge \mathit{Total}\ (\mathit{Pw}\ w)$
begin

A *match* assigns at most one man to each woman, and vice-versa. It is also *individually rational*, i.e., the partners are acceptable to each other. The constant *Field* is the union of the *Domain* and *Range* of a relation.

definition $\mathit{match} :: ('m, 'w)\ \mathit{match} \Rightarrow \mathit{bool}$ **where**
  $\mathit{match}\ \mu \longleftrightarrow \mathit{inj\text{-}on}\ \mathit{fst}\ \mu \wedge \mathit{inj\text{-}on}\ \mathit{snd}\ \mu \wedge \mu \subseteq (\bigcup m.\ \{m\} \times \mathit{Field}\ (\mathit{Pm}\ m)) \cap (\bigcup w.\ \mathit{Field}\ (\mathit{Pw}\ w) \times \{w\})$

A woman *prefers* one man to another if her preference order ranks the former over the latter, and *strictly prefers* him if additionally the latter is not ranked over the former, and similarly for the men.

abbreviation $(\mathit{input})\ \mathit{m\text{-}for}\ w\ \mu \equiv \{m.\ (m, w) \in \mu\}$
abbreviation $(\mathit{input})\ \mathit{w\text{-}for}\ m\ \mu \equiv \{w.\ (m, w) \in \mu\}$

definition $\mathit{m\text{-}prefers} :: 'm \Rightarrow ('m, 'w)\ \mathit{match} \Rightarrow 'w\ \mathit{set}$ **where**
  $\mathit{m\text{-}prefers}\ m\ \mu = \{w' \in \mathit{Field}\ (\mathit{Pm}\ m).\ \forall w{\in}\mathit{w\text{-}for}\ m\ \mu.\ (w, w') \in \mathit{Pm}\ m\}$

definition $\mathit{w\text{-}prefers} :: 'w \Rightarrow ('m, 'w)\ \mathit{match} \Rightarrow 'm\ \mathit{set}$ **where**
  $\mathit{w\text{-}prefers}\ w\ \mu = \{m' \in \mathit{Field}\ (\mathit{Pw}\ w).\ \forall m{\in}\mathit{m\text{-}for}\ w\ \mu.\ (m, m') \in \mathit{Pw}\ w\}$

definition $\mathit{m\text{-}strictly\text{-}prefers} :: 'm \Rightarrow ('m, 'w)\ \mathit{match} \Rightarrow 'w\ \mathit{set}$ **where**
  $\mathit{m\text{-}strictly\text{-}prefers}\ m\ \mu = \{w' \in \mathit{Field}\ (\mathit{Pm}\ m).\ \forall w{\in}\mathit{w\text{-}for}\ m\ \mu.\ (w, w') \in \mathit{Pm}\ m \wedge (w', w) \notin \mathit{Pm}\ m\}$

definition $\mathit{w\text{-}strictly\text{-}prefers} :: 'w \Rightarrow ('m, 'w)\ \mathit{match} \Rightarrow 'm\ \mathit{set}$ **where**
  $\mathit{w\text{-}strictly\text{-}prefers}\ w\ \mu = \{m' \in \mathit{Field}\ (\mathit{Pw}\ w).\ \forall m{\in}\mathit{m\text{-}for}\ w\ \mu.\ (m, m') \in \mathit{Pw}\ w \wedge (m', m) \notin \mathit{Pw}\ w\}$

A couple *blocks* a match μ if both strictly prefer each other to anyone they are matched with in μ.

definition $\mathit{blocks} :: 'm \Rightarrow 'w \Rightarrow ('m, 'w)\ \mathit{match} \Rightarrow \mathit{bool}$ **where**
  $\mathit{blocks}\ m\ w\ \mu \longleftrightarrow w \in \mathit{m\text{-}strictly\text{-}prefers}\ m\ \mu \wedge m \in \mathit{w\text{-}strictly\text{-}prefers}\ w\ \mu$

We say a match is *stable* if there are no blocking couples.

definition $\mathit{stable} :: ('m, 'w)\ \mathit{match} \Rightarrow \mathit{bool}$ **where**
  $\mathit{stable}\ \mu \longleftrightarrow \mathit{match}\ \mu \wedge (\forall m\ w.\ \neg\ \mathit{blocks}\ m\ w\ \mu)$

lemma *stable-match*:
  assumes *stable* μ
  shows *match* μ
using *assms* **unfolding** *stable-def* **by** *blast*

Our goal is to show that for every preference order there is a stable match. Stable matches in this scenario form a lattice, and this proof implicitly adopts the traditional view that men propose and women choose.

The definitions above form the trust basis for this existence theorem; the following are merely part of the proof apparatus, and Isabelle/HOL enforces their soundness with respect to the argument. We will see these concepts again in later developments.

Firstly, a match is *simple* if every woman party to a blocking pair is single. The most obvious such match leaves everyone single.

definition *simple* :: $('m, 'w)$ *match* $\Rightarrow$ *bool* **where**
  simple $\mu \longleftrightarrow$ *match* $\mu \land (\forall\, m\ w.$ *blocks* $m\ w\ \mu \longrightarrow w \notin$ *Range* $\mu)$

lemma *simple-match*:
  assumes *simple* μ
  shows *match* μ
using *assms* **unfolding** *simple-def* **by** *blast*

lemma *simple-ex*:
  $\exists\,\mu.$ *simple* μ
unfolding *simple-def blocks-def match-def* **by** *auto*

Sotomayor observes the following:

lemma *simple-no-single-women-stable*:
  assumes *simple* μ
  assumes $\forall\, w.\ w \in$ *Range* μ — No woman is single
  shows *stable* μ
using *assms* **unfolding** *simple-def stable-def* **by** *blast*

lemma *stable-simple*:
  assumes *stable* μ
  shows *simple* μ
using *assms* **unfolding** *simple-def stable-def* **by** *blast*

Secondly, a *weakly Pareto optimal match for men (among all simple matches)* is one for which there is no other match that all men like as much and some man likes more.

definition *m-weakly-prefers* :: $'m \Rightarrow ('m, 'w)$ *match* $\Rightarrow$ $'w$ *set* **where**
  m-weakly-prefers $m\ \mu = \{w' \in$ *Field* $(Pm\ m).\ \forall\, w{\in}$*w-for* $m\ \mu.\ (w, w') \in Pm\ m\}$

definition *weakly-preferred-by-men* :: $('m, 'w)$ *match* $\Rightarrow ('m, 'w)$ *match* $\Rightarrow$ *bool* **where**
  weakly-preferred-by-men $\mu\ \mu'$
    $\longleftrightarrow (\forall\, m.\ \forall\, w{\in}$*w-for* $m\ \mu.\ \exists\, w'{\in}$*w-for* $m\ \mu'.\ w' \in$ *m-weakly-prefers* $m\ \mu)$

definition *strictly-preferred-by-a-man* :: $('m, 'w)$ *match* $\Rightarrow ('m, 'w)$ *match* $\Rightarrow$ *bool* **where**
  strictly-preferred-by-a-man $\mu\ \mu'$
    $\longleftrightarrow (\exists\, m.\ \exists\, w{\in}$*w-for* $m\ \mu'.\ w \in$ *m-strictly-prefers* $m\ \mu)$

definition *weakly-Pareto-optimal-for-men* :: $('m, 'w)$ *match* $\Rightarrow$ *bool* **where**
  weakly-Pareto-optimal-for-men μ
    $\longleftrightarrow$ *simple* $\mu \land \neg(\exists\,\mu'.$ *simple* $\mu' \land$ *weakly-preferred-by-men* $\mu\ \mu' \land$ *strictly-preferred-by-a-man* $\mu\ \mu')$

We will often provide *introduction rules* for more complex predicates, and sometimes derive these by elementary syntactic manipulations expressed by the *attributes* enclosed in square brackets after a use-mention of a lemma. The **lemmas** command binds a name to the result. To conform with the Isar structured proof language, we use meta-logic ("Pure" in Isabelle terminology) connectives: $\bigwedge$ denotes universal quantification, and $\Longrightarrow$ implication.

lemma *weakly-preferred-by-menI*:
  assumes $\bigwedge m\ w.\ (m, w) \in \mu \Longrightarrow \exists\, w'.\ (m, w') \in \mu' \land w' \in$ *m-weakly-prefers* $m\ \mu$
  shows *weakly-preferred-by-men* $\mu\ \mu'$
using *assms* **unfolding** *weakly-preferred-by-men-def* **by** *blast*

lemmas *simpleI* = *iffD2*[*OF simple-def*, *unfolded conj-imp-eq-imp-imp*, *rule-format*]

lemma *weakly-Pareto-optimal-for-men-simple*:
  assumes *weakly-Pareto-optimal-for-men* μ
  shows *simple* μ
using *assms* **unfolding** *weakly-Pareto-optimal-for-men-def* **by** *simp*

Later we will elide obvious technical lemmas like the following. The more obscure proofs are typically generated automatically by sledgehammer (Blanchette et al. 2016).

lemma *m-weakly-prefers-Pm*:
  assumes *match* μ
  assumes $(m, w) \in \mu$
  shows $w' \in$ *m-weakly-prefers* $m\ \mu \longleftrightarrow (w, w') \in Pm\ m$
using *spec*[*OF Pm-pref*, **where** $x=m$] *assms* **unfolding** *m-weakly-prefers-def match-def preorder-on-def*
by *simp* (*metis* (*no-types*, *opaque-lifting*) *FieldI2 fst-conv inj-on-contraD snd-conv*)

lemma *match-Field*:
  assumes *match* μ
  assumes $(m, w) \in \mu$
  shows $w \in$ *Field* $(Pm\ m)$
    and $m \in$ *Field* $(Pw\ w)$
using *assms* **unfolding** *match-def* **by** *blast+*

lemma *weakly-preferred-by-men-refl*:
  assumes *match* μ
  shows *weakly-preferred-by-men* $\mu\ \mu$
using *assms* **unfolding** *weakly-preferred-by-men-def m-weakly-prefers-def*
by *clarsimp* (*meson Pm-pref m-weakly-prefers-Pm match-Field*(*1*) *preorder-on-def refl-onD*)

Sotomayor, p137 provides an alternative definition of *weakly-preferred-by-men*. The syntax (**is** *?lhs* $\longleftrightarrow$ *pat*) binds the *schematic variables* *?lhs* and *?rhs* to the terms separated by $\longleftrightarrow$.

lemma *weakly-preferred-by-men-strictly-preferred-by-a-man*:
  assumes *match* μ
  assumes *match* μ'
  shows *weakly-preferred-by-men* $\mu\ \mu' \longleftrightarrow \neg$*strictly-preferred-by-a-man* $\mu'\ \mu$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)
proof(*rule iffI*)
  assume *?lhs* **then show** *?rhs*
    unfolding *weakly-preferred-by-men-def strictly-preferred-by-a-man-def*
        m-weakly-prefers-def m-strictly-prefers-def **by** *fastforce*
next
  assume *?rhs* **show** *?lhs*
  proof(*rule weakly-preferred-by-menI*)
    fix $m\ w$ **assume** $(m, w) \in \mu$
    from *assms* ‹*?rhs*› ‹$(m, w) \in \mu$› **obtain** w' **where** *XXX*: $(m, w') \in \mu'$ $(w', w) \in Pm\ m \longrightarrow (w, w') \in Pm\ m$
      unfolding *match-def strictly-preferred-by-a-man-def m-strictly-prefers-def* **by** *blast*
    with *spec*[*OF Pm-pref*, **where** $x=m$] *assms* ‹$(m, w) \in \mu$›
    show $\exists w'.\ (m, w') \in \mu' \wedge w' \in$ *m-weakly-prefers* $m\ \mu$
      unfolding *preorder-on-def total-on-def* **by** (*metis m-weakly-prefers-Pm match-Field*(*1*) *refl-onD*)
  qed
qed

lemma *weakly-Pareto-optimal-for-men-def2*:
  weakly-Pareto-optimal-for-men μ
    $\longleftrightarrow$ *simple* $\mu \wedge (\forall \mu'.$ *simple* $\mu' \wedge$ *strictly-preferred-by-a-man* $\mu\ \mu' \longrightarrow$ *strictly-preferred-by-a-man* $\mu'\ \mu)$
unfolding *weakly-Pareto-optimal-for-men-def simple-def*
by (*meson weakly-preferred-by-men-strictly-preferred-by-a-man*)

Sotomayor claims that the existence of such a weakly Pareto optimal match for men is "guaranteed by the fact that *the set of simple matchings is nonempty* [our *simple-ex* lemma] *and finite and the preferences are transitive.*" The following lemmas express this intuition:

```
lemma trans-finite-has-maximal-elt:
  assumes trans r
  assumes finite (Field r)
  assumes Field r ≠ {}
  shows ∃x∈Field r. (∀y∈Field r. (x, y) ∈ r ⟶ (y, x) ∈ r)
using assms(2,1,3) by induct (auto elim: transE)

lemma weakly-Pareto-optimal-for-men-ex:
  ∃μ. weakly-Pareto-optimal-for-men μ
proof −
  let ?r = {(μ, μ′). simple μ ∧ simple μ′ ∧ weakly-preferred-by-men μ μ′}
  from trans-finite-has-maximal-elt[where r=?r]
  obtain x where x ∈ Field ?r ∀y∈Field ?r. (x, y) ∈ ?r ⟶ (y, x) ∈ ?r
  proof
    from Pm-pref show trans ?r
      unfolding trans-def weakly-preferred-by-men-def m-weakly-prefers-def m-strictly-prefers-def
      by simp (meson order-on-defs(1) transE)
    from simple-ex weakly-preferred-by-men-refl[OF simple-match] show Field ?r ≠ {}
      unfolding Field-def by force
  qed simp-all
  then show ?thesis
    unfolding weakly-Pareto-optimal-for-men-def Field-def
    using simple-match weakly-preferred-by-men-strictly-preferred-by-a-man by auto
qed
```

The main result proceeds by contradiction.

```
lemma weakly-Pareto-optimal-for-men-stable:
  assumes weakly-Pareto-optimal-for-men μ
  shows stable μ
proof(rule ccontr)
  assume ¬stable μ
  from ‹weakly-Pareto-optimal-for-men μ› have simple μ by (rule weakly-Pareto-optimal-for-men-simple)
  from ‹¬stable μ› ‹simple μ› obtain m′ w where blocks m′ w μ and w ∉ Range μ
    unfolding simple-def stable-def by blast+
  — Choose an m that w weakly prefers to any blocking man.
  — We restrict the preference order Pw w to the men who strictly prefer w over their match in μ.
  let ?r = Restr (Pw w) {m. w ∈ m-strictly-prefers m μ}
  from trans-finite-has-maximal-elt[where r=?r]
  obtain m where m ∈ Field ?r ∀m′∈Field ?r. (m, m′) ∈ ?r ⟶ (m′, m) ∈ ?r
  proof
    from Pw-pref show trans ?r
      unfolding preorder-on-def by (blast intro: trans-Restr)
    from Pw-pref ‹blocks m′ w μ› have (m′, m′) ∈ ?r
      unfolding blocks-def w-strictly-prefers-def preorder-on-def by (blast dest: refl-onD)
    then show Field ?r ≠ {} by (metis FieldI2 empty-iff)
  qed simp-all
  with ‹blocks m′ w μ› ‹w ∉ Range μ›
  have blocks m w μ and ∀m′. blocks m′ w μ ∧ (m, m′) ∈ Pw w ⟶ (m′, m) ∈ Pw w
    unfolding blocks-def w-strictly-prefers-def Field-def by auto
  — Construct a new (simple) match containing the blocking pair...
  let ?μ′ = μ − {(m, w′) |w′. True} ∪ {(m, w)}
  — ...and show that it is a Pareto improvement for men over μ.
  have simple ?μ′
  proof(rule simpleI)
    from ‹simple μ› ‹blocks m w μ› show match ?μ′
      unfolding blocks-def match-def simple-def m-strictly-prefers-def w-strictly-prefers-def
      by (safe; clarsimp simp: inj-on-diff; blast)
    fix m′ w′ assume blocks m′ w′ ?μ′
```

```
    from ‹blocks m′ w′ ?μ′› ‹∀ m′. blocks m′ w μ ∧ (m, m′) ∈ Pw w ⟶ (m′, m) ∈ Pw w›
    have w′ ≠ w
      unfolding blocks-def m-strictly-prefers-def w-strictly-prefers-def by auto
    show w′ ∉ Range ?μ′
    proof(cases (m, w′) ∈ μ)
      case True
      from ‹simple μ› ‹blocks m′ w′ ?μ′› ‹w′ ≠ w› ‹(m, w′) ∈ μ›
      show ?thesis
        unfolding simple-def match-def
        by clarsimp (metis (no-types, opaque-lifting) fst-conv inj-on-contraD snd-conv)
    next
      case False
      from Pm-pref ‹blocks m w μ› ‹blocks m′ w′ ?μ′› ‹(m, w′) ∉ μ›
      have blocks m′ w′ μ
        unfolding preorder-on-def blocks-def m-strictly-prefers-def w-strictly-prefers-def
        by simp (metis transE)
      with ‹simple μ› ‹w′ ≠ w› show ?thesis unfolding simple-def by blast
    qed
  qed
  moreover have weakly-preferred-by-men μ ?μ′
  proof(rule weakly-preferred-by-menI)
    fix m′ w′ assume (m′, w′) ∈ μ
    then show ∃ w′. (m′, w′) ∈ ?μ′ ∧ w′ ∈ m-weakly-prefers m′ μ
    proof(cases m′ = m)
      case True
      from ‹blocks m w μ› ‹(m′, w′) ∈ μ› ‹m′ = m› show ?thesis
        unfolding m-weakly-prefers-def blocks-def m-strictly-prefers-def by blast
    next
      case False
      from Pm-pref ‹simple μ› ‹(m′, w′) ∈ μ› ‹m′ ≠ m› show ?thesis
        by clarsimp (meson m-weakly-prefers-Pm match-Field preorder-on-def refl-onD simple-match)
    qed
  qed
  moreover from ‹blocks m w μ› have strictly-preferred-by-a-man μ ?μ′
    unfolding strictly-preferred-by-a-man-def blocks-def by blast
  moreover note ‹weakly-Pareto-optimal-for-men μ›
  ultimately show False
    unfolding weakly-Pareto-optimal-for-men-def by blast
qed

theorem stable-ex:
  ∃ μ. stable μ
using weakly-Pareto-optimal-for-men-stable weakly-Pareto-optimal-for-men-ex by blast
```

We can exit the locale context and later re-enter it.

```
end
```

We *interpret* the locale by supplying constants that instantiate the variables we fixed earlier, and proving that these satisfy the assumptions. In this case we provide concrete preference orders, and by doing so we demonstrate that our theory is non-vacuous. We arbitrarily choose Roth and Sotomayor (1990, Example 2.15) which demonstrates the non-existence of man- or woman-optimal matches if preferences are non-strict. (We define optimality shortly.) The following bunch of types eases the description of this particular scenario.

```
datatype M = M1 | M2 | M3
datatype W = W1 | W2 | W3

lemma M-UNIV: UNIV = set [M1, M2, M3] using M.exhaust by auto
lemma W-UNIV: UNIV = set [W1, W2, W3] using W.exhaust by auto
```

instance *M* :: *finite* **by** *standard* (*simp add*: *M-UNIV*)
instance *W* :: *finite* **by** *standard* (*simp add*: *W-UNIV*)

lemma *M-All*:
  shows $(\forall\, m.\ P\ m) \longleftrightarrow (\forall\, m{\in}set\ [M1,\ M2,\ M3].\ P\ m)$
by (*metis M-UNIV UNIV-I*)

lemma *W-All*:
  shows $(\forall\, w.\ P\ w) \longleftrightarrow (\forall\, w{\in}set\ [W1,\ W2,\ W3].\ P\ w)$
by (*metis W-UNIV UNIV-I*)

primrec *Pm* :: *M* ⇒ *W rel* **where**
  Pm M1 = { (*W1*, *W1*), (*W1*, *W2*), (*W1*, *W3*), (*W2*, *W2*), (*W2*, *W3*), (*W3*, *W3*), (*W3*, *W2*) }
| *Pm M2* = { (*W1*, *W1*), (*W1*, *W2*), (*W2*, *W2*) }
| *Pm M3* = { (*W1*, *W1*), (*W1*, *W3*), (*W3*, *W3*) }

primrec *Pw* :: *W* ⇒ *M rel* **where**
  Pw W1 = { (*M3*, *M3*), (*M3*, *M2*), (*M3*, *M1*), (*M2*, *M2*), (*M2*, *M1*), (*M1*, *M1*) }
| *Pw W2* = { (*M2*, *M2*), (*M2*, *M1*), (*M1*, *M1*) }
| *Pw W3* = { (*M3*, *M3*), (*M3*, *M1*), (*M1*, *M1*) }

lemma *Pm*: *Preorder* (*Pm m*) ∧ *Total* (*Pm m*)
unfolding *preorder-on-def refl-on-def trans-def total-on-def*
by (*cases m*) (*safe*, *auto*)

lemma *Pw*: *Preorder* (*Pw w*) ∧ *Total* (*Pw w*)
unfolding *preorder-on-def refl-on-def trans-def total-on-def*
by (*cases w*) (*safe*, *auto*)

interpretation *Non-Strict*: *StableMarriage Pm Pw*
using *Pm Pw* **by** *unfold-locales blast+*

We demonstrate that there are only two stable matches in this scenario. Isabelle/HOL does not have any special support for these types of model checking problems, so we simply try all combinations of men and women. Clearly this does not scale, and for larger domains we need to be a bit cleverer (see §7).

lemma *Non-Strict-stable1*:
  shows *Non-Strict.stable* {(*M1*, *W2*), (*M2*, *W1*), (*M3*, *W3*)}
unfolding *Non-Strict.stable-def Non-Strict.match-def Non-Strict.blocks-def Non-Strict.m-strictly-prefers-def Non-Strict.w-strictly-prefers-def*
by *clarsimp* (*metis M.exhaust*)

lemma *Non-Strict-stable2*:
  shows *Non-Strict.stable* {(*M1*, *W3*), (*M2*, *W2*), (*M3*, *W1*)}
unfolding *Non-Strict.stable-def Non-Strict.match-def Non-Strict.blocks-def Non-Strict.m-strictly-prefers-def Non-Strict.w-strictly-prefers-def*
by *clarsimp* (*metis M.exhaust*)

lemma *Non-Strict-stable-matches*:
  Non-Strict.stable μ
    $\longleftrightarrow \mu = \{(M1,\ W2),\ (M2,\ W1),\ (M3,\ W3)\}$
    $\lor\ \mu = \{(M1,\ W3),\ (M2,\ W2),\ (M3,\ W1)\}$ (**is** *?lhs* ⟷ *?rhs*)
proof(*rule iffI*)
  assume *?lhs*
  have $\mu \in$ *set* ‘ *set* (*subseqs* (*List.product* [*M1*, *M2*, *M3*] [*W1*, *W2*, *W3*]))
    by (*subst subseqs-powset*; *clarsimp*; *metis M.exhaust W.exhaust*)
  with ‹*?lhs*› **show** *?rhs*
    unfolding *Non-Strict.stable-def Non-Strict.match-def*
    apply (*simp cong*: *INF-cong-simp SUP-cong-simp cong del*: *image-cong-simp*)

```
    apply (elim disjE)
    apply (simp-all cong: INF-cong-simp SUP-cong-simp cong del: image-cong-simp)
    apply (simp-all add: M-All W-All Non-Strict.blocks-def Non-Strict.m-strictly-prefers-def
                Non-Strict.w-strictly-prefers-def cong: INF-cong-simp SUP-cong-simp cong del: image-cong-simp)
    done
next
  assume ?rhs with Non-Strict-stable1 Non-Strict-stable2 show ?lhs by blast
qed
```

So far the only interesting result in this interpretation of *StableMarriage* is the *Non-Strict.stable-ex* theorem, i.e., that there is a stable match. We now add the notion of *optimality* to our locale, and all interpretations will automatically inherit it. Later we will also extend locales by adding new fixed variables and assumptions.

Following Roth and Sotomayor (1990, Definition 2.11), a stable match is *optimal for men* if every man likes it at least as much as any other stable match (and similarly for an *optimal for women* match).

```
context StableMarriage
begin

definition optimal-for-men :: ('m, 'w) match ⇒ bool where
  optimal-for-men μ
    ⟷ stable μ ∧ (∀ μ'. stable μ' ⟶ weakly-preferred-by-men μ' μ)

definition w-weakly-prefers :: 'w ⇒ ('m, 'w) match ⇒ 'm set where
  w-weakly-prefers w μ = {m' ∈ Field (Pw w). ∀ m∈m-for w μ. (m, m') ∈ Pw w}

definition weakly-preferred-by-women :: ('m, 'w) match ⇒ ('m, 'w) match ⇒ bool where
  weakly-preferred-by-women μ μ'
    ⟷ (∀ w. ∀ m∈m-for w μ. ∃ m'∈m-for w μ'. m' ∈ w-weakly-prefers w μ)

definition optimal-for-women :: ('m, 'w) match ⇒ bool where
  optimal-for-women μ
    ⟷ stable μ ∧ (∀ μ'. stable μ' ⟶ weakly-preferred-by-women μ μ')

end
```

We can show that there is no optimal stable match for these preferences:

```
lemma NonStrict-not-optimal:
  assumes Non-Strict.stable μ
  shows ¬Non-Strict.optimal-for-men μ ∧ ¬Non-Strict.optimal-for-women μ
proof −
  from assms[unfolded Non-Strict-stable-matches] show ?thesis
  proof(rule disjE)
    assume μ = {(M1, W2), (M2, W1), (M3, W3)}
    with assms show ?thesis
      unfolding Non-Strict.optimal-for-men-def Non-Strict.weakly-preferred-by-men-def
              Non-Strict.m-weakly-prefers-def Non-Strict.optimal-for-women-def
              Non-Strict.weakly-preferred-by-women-def Non-Strict.w-weakly-prefers-def
              Non-Strict-stable-matches
      by clarsimp (rule conjI; rule exI[where x={(M1, W3), (M2, W2), (M3, W1)}]; blast)
  next
    assume μ = {(M1, W3), (M2, W2), (M3, W1)}
    with assms show ?thesis
      unfolding Non-Strict.optimal-for-men-def Non-Strict.weakly-preferred-by-men-def
              Non-Strict.m-weakly-prefers-def Non-Strict.optimal-for-women-def
              Non-Strict.weakly-preferred-by-women-def Non-Strict.w-weakly-prefers-def
              Non-Strict-stable-matches
      by clarsimp (rule conjI; rule exI[where x={(M1, W2), (M2, W1), (M3, W3)}]; blast)
  qed
qed
```

Sotomayor (1996) remarks that, if the preferences are strict, there is only one weakly Pareto optimal match for men, and that it is man-optimal. (This is the match found by the classic man-proposing deferred acceptance algorithm due to Gale and Shapley (1962).) However she omits a proof that the man-optimal match actually exists under strict preferences.

The easiest way to show this and further results is to exhibit the lattice structure of the stable matches discovered by Conway (see Roth and Sotomayor (1990, Theorem 2.16)), where the men- and women-optimal matches are the extremal points. This suggests looking for a monotonic function whose fixed points are this lattice, which is the essence of the analysis of matching with contracts in §5.

3 Preliminaries

3.1 MaxR: maximum elements of linear orders

We generalize the existing *max* and *Max* functions to work on orders defined over sets. See §4.6 for choice-function related lemmas.

locale *MaxR* =
  fixes *r* :: *'a::finite rel*
  assumes *r-Linear-order*: *Linear-order r*
begin

The basic function chooses the largest of two elements:

definition *maxR* :: $'a \Rightarrow 'a \Rightarrow 'a$ **where**
  maxR x y = (*if* (*x*, *y*) ∈ *r then y else x*)

We hoist this to finite sets using the *Finite-Set.fold* combinator. For code generation purposes it seems inevitable that we need to fuse the fold and filter into a single total recursive definition.

definition *MaxR-f* :: $'a \Rightarrow 'a$ *option* $\Rightarrow 'a$ *option* **where**
  MaxR-f x acc = (*if x* ∈ *Field r then Some* (*case acc of None* ⇒ *x* | *Some y* ⇒ *maxR x y*) *else acc*)

interpretation *MaxR-f*: *comp-fun-idem MaxR-f*

definition *MaxR-opt* :: *'a set* ⇒ *'a option* **where**
  MaxR-opt-eq-fold': *MaxR-opt A* = *Finite-Set.fold MaxR-f None A*
end

interpretation *MaxR-empty*: *MaxR* {}
by *unfold-locales simp*

interpretation *MaxR-singleton*: *MaxR* {(*x*,*x*)} **for** *x*
by *unfold-locales simp*

lemma *MaxR-r-domain* [*iff*]:
  assumes *MaxR r*
  shows *MaxR* (*Restr r A*)
using *assms Linear-order-Restr* **unfolding** *MaxR-def* **by** *blast*

3.2 Linear orders from lists

Often the easiest way to specify a concrete linear order is with a list. Here these run from greatest to least.

primrec *linord-of-listP* :: *'a* ⇒ *'a* ⇒ *'a list* ⇒ *bool* **where**
  linord-of-listP x y [] ⟷ *False*
| *linord-of-listP x y* (*z* # *zs*) ⟷ (*z* = *y* ∧ *x* ∈ *set* (*z* # *zs*)) ∨ *linord-of-listP x y zs*

definition *linord-of-list* :: *'a list* ⇒ *'a rel* **where**
  linord-of-list xs ≡ {(*x*, *y*). *linord-of-listP x y xs*}

lemma *linord-of-list-Linear-order*:
  assumes *distinct xs*
  assumes *ys* = *set xs*
  shows *linear-order-on ys* (*linord-of-list xs*)

Every finite linear order is generated by a list.

lemma *linear-order-on-list*:
  assumes *linear-order-on ys r*
  assumes *ys* = *Field r*
  assumes *finite ys*
  shows $\exists!xs.\ r = \textit{linord-of-list}\ xs \wedge \textit{distinct}\ xs \wedge \textit{set}\ xs = ys$

4 Choice Functions

We now develop a few somewhat general results about choice functions, following Border (2012); Moulin (1985); Sen (1970). Hansson and Grüne-Yanoff (2012) provide some philosophical background on this topic. While this material is foundational to the story we tell about stable matching, it is perhaps best skipped over on a first reading.

The game here is to study conditions on functions that yield acceptable choices from a given set of alternatives drawn from some universe (a set, often a type in HOL). We adopt the Isabelle convention of attaching the suffix *on* to predicates that are defined on subsets of their types.

type-synonym *'a cfun* = *'a set* ⇒ *'a set*

Most results require that the choice function yield a subset of its argument:

definition *f-range-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
  f-range-on A f ⟷ ($\forall B \subseteq A.\ f\ B \subseteq B$)

abbreviation *f-range* :: *'a cfun* ⇒ *bool* **where**
  f-range ≡ *f-range-on UNIV*

Economists typically assume that the universe is finite, and f is *decisive*, i.e., yields non-empty sets when given non-empty sets.

definition *decisive-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
  decisive-on A f ⟷ ($\forall B \subseteq A.\ B \neq \{\} \longrightarrow f\ B \neq \{\}$)

abbreviation *decisive* :: *'a cfun* ⇒ *bool* **where**
  decisive ≡ *decisive-on UNIV*

Often we can mildly generalise existing results by not requiring that f be *decisive*, and by dropping the finiteness hypothesis. We make essential use of the former generalization in §5.

Some choice functions, such as those arising from linear orders (§4.6), are *resolute*: these always yield a single choice.

definition *resolute-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
  resolute-on A f ⟷ ($\forall B \subseteq A.\ B \neq \{\} \longrightarrow (\exists a.\ f\ B = \{a\})$)

abbreviation *resolute* :: *'a cfun* ⇒ *bool* **where**
  resolute ≡ *resolute-on UNIV*

lemma *resolute-on-decisive-on*:
  assumes *resolute-on A f*
  shows *decisive-on A f*

Often we talk about the choices that are rejected by f:

abbreviation *Rf* :: *'a cfun* ⇒ *'a cfun* **where**
Rf f X ≡ *X* − *f X*

Typically there are many (almost-)equivalent formulations of each property in the literature. We try to formulate our rules in terms of the most general of these.

4.1 The *substitutes* condition, AKA *independence of irrelevant alternatives* AKA *Chernoff*

Loosely speaking, the *substitutes* condition asserts that an alternative that is rejected from A shall remain rejected when there is "increased competition," i.e., from all sets that contain A.

Hatfield and Milgrom (2005) define this property as simply the monotonicity of *Rf*. Aygün and Sönmez (2012b) instead use the complicated condition shown here. Condition α, due to Sen (1970, p17, see below), is the most general and arguably the most perspicuous.

definition *substitutes-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
substitutes-on A f ⟷ ¬(∃ *B*⊆*A*. ∃ *a b*. {*a*, *b*} ⊆ *A* − *B* ∧ *b* ∉ *f* (*B* ∪ {*b*}) ∧ *b* ∈ *f* (*B* ∪ {*a*, *b*}))

abbreviation *substitutes* :: *'a cfun* ⇒ *bool* **where**
substitutes ≡ *substitutes-on UNIV*

lemma *substitutes-on-def2*[*simplified*]:
substitutes-on A f ⟷ (∀ *B*⊆*A*. ∀ *a*∈*A*. ∀ *b*∈*A*. *b* ∉ *f* (*B* ∪ {*b*}) ⟶ *b* ∉ *f* (*B* ∪ {*a*, *b*}))

lemma *substitutes-on-union*:
assumes *a* ∉ *f* (*B* ∪ {*a*})
assumes *substitutes-on* (*A* ∪ *B* ∪ {*a*}) *f*
assumes *finite A*
shows *a* ∉ *f* (*A* ∪ *B* ∪ {*a*})

lemma *substitutes-on-antimono*:
assumes *substitutes-on B f*
assumes *A* ⊆ *B*
shows *substitutes-on A f*

The equivalence with the monotonicity of alternative-rejection requires a finiteness constraint.

lemma *substitutes-on-Rf-mono-on*:
assumes *substitutes-on A f*
assumes *finite A*
shows *mono-on* (*Pow A*) (*Rf f*)

lemma *Rf-mono-on-substitutes*:
assumes *mono-on* (*Pow A*) (*Rf f*)
shows *substitutes-on A f*

The above substitutes condition is equivalent to the *independence of irrelevant alternatives*, AKA condition α due to Sen (1970). Intuitively if a is chosen from a set A, then it must be chosen from every subset of A that it belongs to. Note the lack of finiteness assumptions here.

definition *iia-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
iia-on A f ⟷ (∀ *B*⊆*A*. ∀ *C*⊆*B*. ∀ *a*∈*C*. *a* ∈ *f B* ⟶ *a* ∈ *f C*)

abbreviation *iia* :: *'a cfun* ⇒ *bool* **where**
iia ≡ *iia-on UNIV*

lemma *Rf-mono-on-iia-on*:
shows *mono-on* (*Pow A*) (*Rf f*) ⟷ *iia-on A f*

lemma *Rf-mono-iia*:

shows *mono* (*Rf f*) $\longleftrightarrow$ *iia f*

lemma *substitutes-iia*:
assumes *finite A*
shows *substitutes-on A f* $\longleftrightarrow$ *iia-on A f*

One key result is that the choice function must be idempotent if it satisfies *iia* or any of the equivalent conditions.

lemma *iia-f-idem*:
assumes *f-range-on A f*
assumes *iia-on A f*
assumes $B \subseteq A$
shows $f\ (f\ B) = f\ B$

Hatfield and Milgrom (2005, p914, bottom right) claim that the *substitutes* condition coincides with the *substitutable preferences* condition for the college admissions problem of Roth and Sotomayor (1990, Definition 6.2), which is similar to *iia*:

definition *substitutable-preferences-on* :: *'a set* $\Rightarrow$ *'a cfun* $\Rightarrow$ *bool* **where**
substitutable-preferences-on A f $\longleftrightarrow$ $(\forall B{\subseteq}A.\ \forall a{\in}B.\ \forall b{\in}B.\ a \neq b \wedge a \in f\ B \longrightarrow a \in f\ (B - \{b\}))$

lemma *substitutable-preferences-on-substitutes-on*:
shows *substitutable-preferences-on A f* $\longleftrightarrow$ *substitutes-on A f* (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

Moulin (1985, p152) defines an equivalent *Chernoff* condition. Intuitively this captures the idea that "a best choice in some issue [set of alternatives] is still best if the issue shrinks."

definition *Chernoff-on* :: *'a set* $\Rightarrow$ *'a cfun* $\Rightarrow$ *bool* **where**
Chernoff-on A f $\longleftrightarrow$ $(\forall B{\subseteq}A.\ \forall C{\subseteq}B.\ f\ B \cap C \subseteq f\ C)$

abbreviation *Chernoff* :: *'a cfun* $\Rightarrow$ *bool* **where**
Chernoff $\equiv$ *Chernoff-on UNIV*

lemmas *Chernoff-onI* = *iffD2*[*OF Chernoff-on-def*, *rule-format*]
lemmas *Chernoff-def* = *Chernoff-on-def*[**where** *A*=*UNIV*, *simplified*]

lemma *Chernoff-on-iia-on*:
shows *Chernoff-on A f* $\longleftrightarrow$ *iia-on A f*

lemma *Chernoff-on-union*:
assumes *Chernoff-on A f*
assumes *f-range-on A f*
assumes $B \subseteq A\ \ C \subseteq A$
shows $f\ (B \cup C) \subseteq f\ B \cup f\ C$

Moulin (1985, p159) states a series of equivalent formulations of the *Chernoff* condition. He also claims that these hold if the two sets are disjoint.

lemma *Chernoff-a*:
assumes *f-range-on A f*
shows *Chernoff-on A f* $\longleftrightarrow$ $(\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) \subseteq f\ B \cup C)$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

lemma *Chernoff-b*: — essentially the converse of *Chernoff-on-union*
assumes *f-range-on A f*
shows *Chernoff-on A f* $\longleftrightarrow$ $(\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) \subseteq f\ B \cup f\ C)$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

lemma *Chernoff-c*:
assumes *f-range-on A f*

shows $\mathit{Chernoff\text{-}on}\ A\ f \longleftrightarrow (\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) \subseteq f\ (f\ B \cup C))$ (**is** $?lhs \longleftrightarrow ?rhs$)

lemma *Chernoff-d*:
assumes *f-range-on* $A\ f$
shows $\mathit{Chernoff\text{-}on}\ A\ f \longleftrightarrow (\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) \subseteq f\ (f\ B \cup f\ C))$ (**is** $?lhs \longleftrightarrow ?rhs$)

4.2 The *irrelevance of rejected contracts* condition AKA *consistency* AKA *Aizerman*

Aygün and Sönmez (2012b, §4) propose to repair the results of Hatfield and Milgrom (2005) by imposing the *irrelevance of rejected contracts* (IRC) condition. Intuitively this requires the choice function f to ignore unchosen alternatives.

definition *irc-on* :: $'a\ set \Rightarrow 'a\ cfun \Rightarrow bool$ **where**
$\mathit{irc\text{-}on}\ A\ f \longleftrightarrow (\forall B{\subseteq}A.\ \forall a{\in}A.\ a \notin f\ (B \cup \{a\}) \longrightarrow f\ (B \cup \{a\}) = f\ B)$

abbreviation *irc* :: $'a\ cfun \Rightarrow bool$ **where**
$\mathit{irc} \equiv \mathit{irc\text{-}on}\ UNIV$

lemma *irc-on-discard*:
assumes *irc-on* $A\ f$
assumes *finite* C
assumes $B \cup C \subseteq A$
assumes $f\ (B \cup C) \cap C = \{\}$
shows $f\ (B \cup C) = f\ B$

An equivalent condition is called *consistency* by some (Chambers and Yenmez (2013, Definition 2), Fleiner (2002, Equation (14))). Like *iia*, this formulation generalizes to infinite universes.

definition *consistency-on* :: $'a\ set \Rightarrow 'a\ cfun \Rightarrow bool$ **where**
$\mathit{consistency\text{-}on}\ A\ f \longleftrightarrow (\forall B{\subseteq}A.\ \forall C{\subseteq}B.\ f\ B \subseteq C \longrightarrow f\ B = f\ C)$

abbreviation *consistency* :: $'a\ cfun \Rightarrow bool$ **where**
$\mathit{consistency} \equiv \mathit{consistency\text{-}on}\ UNIV$

lemma *irc-on-consistency-on*:
assumes *irc-on* $A\ f$
assumes *finite* A
shows *consistency-on* $A\ f$

lemma *consistency-on-irc-on*:
assumes *f-range-on* $A\ f$
assumes *consistency-on* $A\ f$
shows *irc-on* $A\ f$

These conditions imply that f is idempotent:

lemma *consistency-on-f-idem*:
assumes *f-range-on* $A\ f$
assumes *consistency-on* $A\ f$
assumes $B \subseteq A$
shows $f\ (f\ B) = f\ B$

Moulin (1985, p154) defines a similar but weaker property he calls *Aizerman*:

definition *Aizerman-on* :: $'a\ set \Rightarrow 'a\ cfun \Rightarrow bool$ **where**
$\mathit{Aizerman\text{-}on}\ A\ f \longleftrightarrow (\forall B{\subseteq}A.\ \forall C{\subseteq}B.\ f\ B \subseteq C \longrightarrow f\ C \subseteq f\ B)$

abbreviation *Aizerman* :: $'a\ cfun \Rightarrow bool$ **where**

Aizerman ≡ *Aizerman-on UNIV*

lemma *consistency-on-Aizerman-on*:
assumes *consistency-on A f*
shows *Aizerman-on A f*

The converse requires f to be idempotent (Moulin 1985, p157):

lemma *Aizerman-on-idem-on-consistency-on*:
assumes *Aizerman-on A f*
assumes $\forall B \subseteq A.\ f\ (f\ B) = f\ B$
shows *consistency-on A f*

4.3 The *law of aggregate demand* condition aka *size monotonicity*

Hatfield and Milgrom (2005, §III) impose the *law of aggregate demand* (aka *size monotonicity*) to obtain the rural hospitals theorem (§5.6). It captures the following intuition:

> [...] Roughly, this law states that as the price falls, agents should demand more of a good. Here, price falls correspond to more contracts being available, and more demand corresponds to taking on (weakly) more contracts.

The *card* function takes a finite set into its cardinality (as a natural number).

definition *lad-on* :: *'a set* ⇒ *'a::finite cfun* ⇒ *bool* **where**
$lad\text{-}on\ A\ f \longleftrightarrow (\forall B \subseteq A.\ \forall C \subseteq B.\ card\ (f\ C) \leq card\ (f\ B))$

abbreviation *lad* :: *'a::finite cfun* ⇒ *bool* **where**
lad ≡ *lad-on UNIV*

This definition is identical amongst Hatfield and Milgrom (2005, §III), Fleiner (2002, (20)), and Aygün and Sönmez (2012b, Definition 4).

Aygün and Sönmez (2012b, §5, Proposition 1) show that *substitutes* and *lad* imply *irc*, which therefore rescues many results in the matching-with-contracts literature.

lemma *lad-on-substitutes-on-irc-on*:
assumes *f-range-on A f*
assumes *substitutes-on A f*
assumes *lad-on A f*
shows *irc-on A f*

The converse does not hold.

4.4 The *expansion* condition

According to Moulin (1985, p152), a choice function satifies *expansion* if an alternative chosen from two sets is also chosen from their union.

definition *expansion-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
$expansion\text{-}on\ A\ f \longleftrightarrow (\forall B \subseteq A.\ \forall C \subseteq A.\ f\ B \cap f\ C \subseteq f\ (B \cup C))$

abbreviation *expansion* :: *'a cfun* ⇒ *bool* **where**
expansion ≡ *expansion-on UNIV*

Condition γ due to Sen (1971) generalizes *expansion* to collections of sets of choices.

definition *expansion-gamma-on* :: *'a set* ⇒ *'a set set* ⇒ *'a cfun* ⇒ *bool* **where**
$expansion\text{-}gamma\text{-}on\ A\ As\ f \longleftrightarrow (\bigcup As \subseteq A \wedge As \neq \{\} \longrightarrow (\bigcap A \in As.\ f\ A) \subseteq f\ (\bigcup As))$

definition *expansion-gamma* :: *'a set set* ⇒ *'a cfun* ⇒ *bool* **where**
expansion-gamma ≡ *expansion-gamma-on UNIV*

lemma *expansion-gamma-expansion*:
assumes ∀ *As*. *expansion-gamma-on A As f*
shows *expansion-on A f*

lemma *expansion-expansion-gamma*:
assumes *expansion-on A f*
assumes *finite As*
shows *expansion-gamma-on A As f*

The *expansion* condition plays a major role in the study of the *rationalizability* of choice functions, which we explore next.

4.5 Axioms of revealed preference

We digress from our taxonomy of conditions on choice functions to discuss *rationalizability*. A choice function is *rationalizable* if there exists some binary relation that generates it, typically by taking the *greatest* or *maximal* elements of the given set of alternatives:

definition *greatest* :: *'a rel* ⇒ *'a cfun* **where**
greatest r X = {*x*∈*X*. ∀ *y*∈*X*. (*y*, *x*) ∈ *r*}

definition *maximal* :: *'a rel* ⇒ *'a cfun* **where**
maximal r X = {*x*∈*X*. ∀ *y*∈*X*. ¬(*x*, *y*) ∈ *r*}

lemma (**in** *MaxR*) *greatest*:
shows *set-option* (*MaxR-opt X*) = *greatest r* (*X* ∩ *Field r*)

Note that *greatest* requires the relation to be reflexive and total, and *maximal* requires it to be irreflexive, for the choice functions to ever yield non-empty sets.

This game of uncovering the preference relations (if any) underlying a choice function goes by the name of *revealed preference*. (In contrast, later we show how these conditions guarantee the existence of stable many-to-one matches.) See Moulin (1985) and Border (2012) for background, intuition and critique, and Sen (1971) for further classical results and proofs.

We adopt the following notion here:

definition *rationalizes-on* :: *'a set* ⇒ *'a cfun* ⇒ *'a rel* ⇒ *bool* **where**
rationalizes-on A f r ⟷ (∀ *B*⊆*A*. *f B* = *greatest r B*)

abbreviation *rationalizes* :: *'a cfun* ⇒ *'a rel* ⇒ *bool* **where**
rationalizes ≡ *rationalizes-on UNIV*

In words, relation r rationalizes the choice function f over universe A if $f\ B$ picks out the *greatest* elements of $B \subseteq A$ with respect to r. At this point r can be any relation that does the job, but soon enough we will ask that it satisfy some familiar ordering properties.

The analysis begins by determining under what constraints f can be rationalized, continues by establishing some properties of all rationalizable choice functions, and concludes by considering what it takes to establish stronger properties.

Following Border (2012, §5, Definition 2) and Sen (1971, Definition 2), we can generate the *revealed weakly preferred* relation for the choice function f:

definition *rwp-on* :: *'a set* ⇒ *'a cfun* ⇒ *'a rel* **where**
rwp-on A f = {(*x*, *y*). ∃ *B*⊆*A*. *x* ∈ *B* ∧ *y* ∈ *f B*}

abbreviation *rwp* :: *'a cfun* ⇒ *'a rel* **where**
rwp ≡ *rwp-on UNIV*

lemma *rwp-on-refl-on*:
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *refl-on A* (*rwp-on A f*)

In words, if it is ever possible that $x \in B$ is available and f B chooses y, then y is taken to always be at least as good as x.

The *V-axiom* asserts that whatever is revealed to be at least as good as anything else on offer is chosen:

definition *V-axiom-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
V-axiom-on A f ⟷ (∀ *B*⊆*A*. ∀ *y*∈*B*. (∀ *x* ∈ *B*. (*x*, *y*) ∈ *rwp-on A f*) ⟶ *y* ∈ *f B*)

abbreviation *V-axiom* :: *'a cfun* ⇒ *bool* **where**
V-axiom ≡ *V-axiom-on UNIV*

This axiom characterizes rationality; see Border (2012, Theorem 7). Sen (1971, §3) calls a decisive choice function that satisfies *V-axiom normal.*

lemma *rationalizes-on-f-range-on-V-axiom-on*:
assumes *rationalizes-on A f r*
shows *f-range-on A f*
and *V-axiom-on A f*

lemma *f-range-on-V-axiom-on-rationalizes-on*:
assumes *f-range-on A f*
assumes *V-axiom-on A f*
shows *rationalizes-on A f* (*rwp-on A f*)

theorem *V-axiom-on-rationalizes-on*:
shows (*f-range-on A f* ∧ *V-axiom-on A f*) ⟷ (∃ *r*. *rationalizes-on A f r*)

We could also ask that f be determined directly by how it behaves on pairs (Sen (1971), Moulin (1985, p151)), which turns out to be equivalent:

definition *rationalizable-binary-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
rationalizable-binary-on A f ⟷ (∀ *B*⊆*A*. *f B* = {*y* ∈ *B*. ∀ *x*∈*B*. *y* ∈ *f* {*x*, *y*}})

abbreviation *rationalizable-binary* :: *'a cfun* ⇒ *bool* **where**
rationalizable-binary ≡ *rationalizable-binary-on UNIV*

theorem *V-axiom-realizable-binary*:
assumes *f-range-on A f*
shows *V-axiom-on A f* ⟷ *rationalizable-binary-on A f*

All rationalizable choice functions satisfy *iia* and *expansion* (Sen (1971), Moulin (1985, p152)).

lemma *rationalizable-binary-on-iia-on*:
assumes *f-range-on A f*
assumes *rationalizable-binary-on A f*
shows *iia-on A f*

lemma *rationalizable-binary-on-expansion-on*:
assumes *f-range-on A f*
assumes *rationalizable-binary-on A f*
shows *expansion-on A f*

The converse requires the set of alternatives to be finite, and moreover fails if the choice function is not *decisive*.

lemma *rationalizable-binary-on-converse*:
  fixes f :: *'a::finite cfun*
  assumes *f-range-on A f*
  assumes *decisive-on A f*
  assumes *iia-on A f*
  assumes *expansion-on A f*
  shows *rationalizable-binary-on A f*

That settles the issue of existence, but it is not clear that the relation is really "rational" (for instance, *rwp-on A f* need not be transitive). Therefore the analysis continues by further constraining the choice function so that it is rationalized by familiar ordering relations.

For instance, the following shows that the *axioms of revealed preference* are rationalized by total preorders (Sen 1971, Definitions 8 and 13)[1]. These are alo equivalent to some congruence axioms due to Samuelson (Border 2012). We define x to be *strictly revealed-preferred to* y if there is a situation where both are on offer and only y is chosen:

definition *rsp-on* :: *'a set* ⇒ *'a cfun* ⇒ *'a rel* **where** — (Sen 1971, Definition 8)
  $rsp\text{-}on\ A\ f = \{(x,\ y).\ \exists B{\subseteq}A.\ x \in Rf\ f\ B \wedge y \in f\ B\}$

abbreviation *rsp* :: *'a cfun* ⇒ *'a rel* **where**
  $rsp \equiv rsp\text{-}on\ UNIV$

This relation is typically denoted by P, for strict preference. The not-worse-than relation R is recovered by:

definition *rspR-on* :: *'a set* ⇒ *'a cfun* ⇒ *'a rel* **where** — (Sen 1971, Definition 9)
  $rspR\text{-}on\ A\ f = \{(x,\ y).\ \{x,\ y\} \subseteq A \wedge (y,\ x) \notin rsp\text{-}on\ A\ f\}$

abbreviation *rspR* :: *'a cfun* ⇒ *'a rel* **where**
  $rspR \equiv rspR\text{-}on\ UNIV$

Sen (1971, p309) defines the *weak axiom of revealed preference* (WARP) as follows:

definition *warp-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
  $warp\text{-}on\ A\ f \longleftrightarrow (\forall (x,\ y){\in}rsp\text{-}on\ A\ f.\ (y,\ x) \notin rwp\text{-}on\ A\ f)$

abbreviation *warp* :: *'a cfun* ⇒ *bool* **where**
  $warp \equiv warp\text{-}on\ UNIV$

The *strong axiom of revealed preference* (SARP) is essentially the transitive closure of *warp* (Sen 1971, p309):

definition *sarp-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
  $sarp\text{-}on\ A\ f \longleftrightarrow (\forall (x,\ y){\in}(rsp\text{-}on\ A\ f)^+.\ (y,\ x) \notin rwp\text{-}on\ A\ f)$

abbreviation *sarp* :: *'a cfun* ⇒ *bool* **where**
  $sarp \equiv sarp\text{-}on\ UNIV$

lemma *sarp-on-warp-on*: — Sen (1970, T.3 part)
  assumes *sarp-on A f*
  shows *warp-on A f*

lemma *rsp-on-irrefl*:
  $A \neq \{\} \Longrightarrow irrefl\ (rsp\text{-}on\ A\ f)$

For decisive choice functions, *warp* implies *sarp*. We show this following Sen (1971), via the *weak congruence axiom* (WCA): if f chooses x from some set B and y is revealed to be weakly preferred, then f must choose y from B as well.

[1] For Sen (1970, p9), an ordering is complete (total), reflexive, and transitive. Alternative names are: complete pre-ordering, complete quasi-ordering, and weak ordering.

definition *wca-on* :: *'a set* ⇒ *'a cfun* ⇒ *bool* **where**
wca-on A f ⟷ (∀ (*x*, *y*)∈*rwp-on A f*. ∀ *B*⊆*A*. *x* ∈ *f B* ∧ *y* ∈ *B* ⟶ *y* ∈ *f B*)

abbreviation *wca* :: *'a cfun* ⇒ *bool* **where**
wca ≡ *wca-on UNIV*

Decisive choice functions that satisfy *wca* are rationalized by total preorders, in particular *rwp*, and the converse obtains if they are normal.

lemma *wca-on-V-axiom-on*:
assumes *wca-on A f*
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *V-axiom-on A f*

lemma *wca-on-total-on*:
assumes *wca-on A f*
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *total-on A* (*rwp-on A f*)

lemma *rwp-on-trans*:
assumes *wca-on A f*
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *trans* (*rwp-on A f*)

lemma *wca-on-V-axiom-on-preorder-on*: — Sen (1970, T.1, T.3 part)
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *wca-on A f* ⟷ *V-axiom-on A f* ∧ *preorder-on A* (*rwp-on A f*) ∧ *total-on A* (*rwp-on A f*)

lemma *wca-on-rwp-on-rspR-on*: — Sen (1970, T.2)
assumes *wca-on A f*
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *rwp-on A f* = *rspR-on A f*

lemma *rwp-on-rspR-on-wca-on*: — Sen (1970, T.2)
assumes *rwp-on A f* = *rspR-on A f*
shows *wca-on A f*

lemma *wca-on-warp-on*: — Sen (1970, T.3 part)
shows *wca-on A f* ⟷ *warp-on A f*

lemma *warp-on-sarp-on*: — Sen (1970, T.3 part)
assumes *warp-on A f*
assumes *f-range-on A f*
assumes *decisive-on A f*
shows *sarp-on A f*
proof(*rule sarp-onI*)
from ‹*warp-on A f*› **have** *wca-on A f* **unfolding** *wca-on-warp-on* **.**
then have *XXX*: *rwp-on A f* = *rspR-on A f*
and *YYY*: *preorder-on A* (*rspR-on A f*)
and *ZZZ*: *total-on A* (*rspR-on A f*)
fix *a b* **assume** (*a*, *b*) ∈ (*rsp-on A f*)$^+$
then have {*a*, *b*} ⊆ *A* **and** (*b*, *a*) ∉ *rspR-on A f*
proof(*induct a b*)

```
    case (r-into-trancl a b)
    { case 1 from r-into-trancl rsp-on-range[OF assms(2)] show ?case by blast }
    { case 2 from r-into-trancl show ?case by (simp add: rspR-on-def) }
  next
    case (trancl-into-trancl a b c)
    { case 1 from trancl-into-trancl rsp-on-range[OF assms(2)] show ?case by blast }
    { case 2 from trancl-into-trancl rsp-on-range[OF assms(2)] YYY ZZZ show ?case
        unfolding total-on-def preorder-on-def
        by clarsimp (metis (no-types, lifting) case-prodD mem-Collect-eq rspR-on-def transD) }
  qed
  with XXX show (b, a) ∉ rwp-on A f by simp
qed
```

The *decisive* constraint here is necessary: consider a Condorcet cycle over $\{x, y, z\}$: forcing $f\ \{x, y, z\}$ to be non-empty resolves this.

Sen (1971) proves that these and other conditions on choice functions are equivalent (under the *decisive* hypothesis).

4.5.1 The *strong axiom of revealed preference* ala Aygün and Sönmez (2012b)

Aygün and Sönmez (2012b, §6) adopt a different definition for a *strong axiom of revealed preference* and show that it holds for all choice functions that satisfy *iia* and *consistency*.

```
abbreviation nth-mod :: 'a list ⇒ nat ⇒ 'a (infixl ‹!%› 100) where
  xs !% i ≡ xs ! (i mod length xs)

definition mwc-sarp :: 'a cfun ⇒ bool where
  mwc-sarp f ⟷
    ¬(∃Xs. length Xs > 1 ∧ distinct (map f Xs) ∧ (∀i. f (Xs!%i) ⊂ Xs!%i ∩ Xs!%(i+1)))

lemma iia-consistency-mwc-sarp:
  assumes f-range f
  assumes iia f — substitutes
  assumes consistency f — irc
  shows mwc-sarp f
proof(rule mwc-sarpI)
  fix Xs
  assume LLL: length Xs > 1
     and EEE: distinct (map f Xs)
     and AAA: ∀i. f (Xs!%i) ⊂ Xs!%i ∩ Xs!%(i+1)
  have 6: f (⋃(set Xs)) ⊆ (⋂X∈set Xs. f X)
  proof −
    have 4: x ∉ f (⋃(set Xs)) if x ∈ ⋃(set Xs) − (⋃X∈set Xs. f X) for x
      using that ‹iia f› unfolding iia-on-def by simp blast
    have 5: x ∉ f (⋃(set Xs)) if x ∈ (⋃X∈set Xs. f X) − (⋂X∈set Xs. f X) for x
    proof −
      from that obtain j k where x ∈ f (Xs ! j) x ∉ f (Xs ! k) j < length Xs k < length Xs
        by (clarsimp simp: in-set-conv-nth)
      with AAA LLL ex-least-nat-le[where n=k + length Xs − j and P=λi. x ∉ f (Xs !% (i + j))]
      obtain i where x ∈ f (Xs !% i) − f (Xs !% (i+1))
      with AAA have x ∈ Rf f (Xs!%(i+1)) by auto
      with LLL show x ∉ f (⋃(set Xs))
        using ‹iia f› unfolding iia-on-def by clarsimp (meson Suc-lessD Sup-upper mod-less-divisor nth-mem)
    qed
    from 4 5 have x ∉ f (⋃(set Xs)) if x ∈ (⋃(set Xs)) − (⋂X∈set Xs. f X) for x
      using that by blast
    with ‹f-range f› show ?thesis by (blast dest: f-range-onD)
  qed
  moreover have ∀i. (⋂X∈set Xs. f X) ⊂ f (Xs!%i)
  proof −
```

from ‹*f-range f*› *LLL* **have** $\bigcap$(*f* ‘ *set Xs*) ⊆ *Xs* ! *1*
using *nth-mem f-range-onD* **by** *fastforce*
with ‹*consistency f*› *LLL 6* **have** *f4*: *f* ($\bigcup$(*set Xs*)) = *f* (*Xs* ! *1*)
by − (*rule consistencyD*[**where** *f*=*f*], *force*+)
with ‹*f-range f*› *LLL 6* **have** *f* (*Xs* ! *1*) ⊆ *Xs* ! *0*
using *f-range-onD* **by** (*metis INT-lower One-nat-def Suc-lessD subset-trans nth-mem top.extremum*)
with ‹*consistency f*› *EEE LLL f4* **show** *?thesis*
by (*metis One-nat-def Suc-lessD Sup-upper consistencyD length-map nth-eq-iff-index-eq nth-map nth-mem zero-neq-one*)
qed
moreover have ∀ *i*. *f* (*Xs*!%*i*) = *f* ($\bigcup$(*set Xs*))
proof −
from *AAA* **have** ∀ *i*. *f* (*Xs*!%*i*) ⊆ *Xs*!%*i* **by** *auto*
moreover from *LLL* **have** ∀ *i*. *Xs*!%*i* ⊆ $\bigcup$(*set Xs*)
by (*metis One-nat-def Suc-lessD Sup-upper mod-less-divisor nth-mem*)
moreover note *6* ‹∀ *i*. ($\bigcap$*X*∈*set Xs*. *f X*) ⊂ *f* (*Xs* !% *i*)›
ultimately show ∀ *i*. *f* (*Xs*!%*i*) = *f* ($\bigcup$(*set Xs*))
by − (*clarsimp*; *rule consistencyD*[*OF* ‹*consistency f*›, *symmetric*]; *meson dual-order.trans psubsetE*)
qed
ultimately show *False* **by** *force*
qed

4.6 Choice functions arising from linear orders

An obvious way to construct a choice function is to derive one from a linear order, i.e., a list of strict preferences. We allow such rankings to omit some alternatives, which means the resulting function is not decisive.

We work with a finite universe here.

locale *linear-cf* =
fixes *r* :: *'a*::*finite rel*
fixes *linear-cf* :: *'a cfun*
assumes *r-linear*: *Linear-order r*
assumes *linear-cf-def*: *linear-cf X* ≡ *set-option* (*MaxR.MaxR-opt r X*)
begin

interpretation *MaxR*: *MaxR r* **by** *unfold-locales* (*rule r-linear*)

lemma *range*:
shows *linear-cf X* ⊆ *X* ∩ *Field r*

lemmas *range'* = *rev-subsetD*[*OF* - *range*, *of x*] **for** *x*

lemma *singleton*:
shows *x* ∈ *linear-cf X* ⟷ *linear-cf X* = {*x*}

lemma *subset*:
assumes *linear-cf Y* ⊆ *X*
assumes *X* ⊆ *Y*
shows *linear-cf Y* = *linear-cf X*

lemma *union*:
shows *linear-cf* (*X* ∪ *Y*) = (*if linear-cf X* = {} *then linear-cf Y else if linear-cf Y* = {} *then linear-cf X else* {*MaxR.maxR x y* |*x y*. *x* ∈ *linear-cf X* ∧ *y* ∈ *linear-cf Y*})

lemma *mono*:
assumes *x* ∈ *linear-cf X*
shows ∃ *y* ∈ *linear-cf* (*X* ∪ *Y*). (*x*, *y*) ∈ *r*

lemmas *greatest* = *MaxR.greatest*[*folded linear-cf-def*]

lemma *preferred*:
  assumes $(x, y) \in r$
  assumes $x \in$ *linear-cf* X
  assumes $y \in X$
  shows $y = x$

lemma *card-le*:
  shows *card* (*linear-cf* X) $\leq$ *1*

lemma *card*:
  shows *card* (*linear-cf* X) = (*if* $X \cap$ *Field* $r = \{\}$ *then 0 else 1*)

lemma *f-range*:
  shows *f-range-on* X *linear-cf*

lemma *domain*:
  shows *linear-cf* ($X \cap$ *Field* r) = *linear-cf* X

lemma *decisive-on*:
  shows *decisive-on* (*Field* r) *linear-cf*

lemma *resolute-on*:
  shows *resolute-on* (*Field* r) *linear-cf*

lemma *Rf-mono-on*:
  shows *mono-on* X (*Rf linear-cf*)

lemmas *iia* = *iffD1*[*OF Rf-mono-on-iia-on Rf-mono-on*]

lemma *Chernoff*:
  shows *Chernoff-on* X *linear-cf*

lemma *irc*:
  shows *irc-on* X *linear-cf*

lemma *consistency*:
  shows *consistency-on* X *linear-cf*

lemma *lad*:
  shows *lad-on* X *linear-cf*

end

4.7 Plott's *path independence* condition

As recognised by Fleiner (2002, §4) and Chambers and Yenmez (2013) in the context of matching with contracts, the *irc* and *substitutes* conditions together are equivalent to *path independence*, a condition introduced to the social choice setting by Plott (1973). Moulin (1985, Lemma 6) ascribes this equivalence result to Aizerman and Malishevski (1981).

definition *path-independent-on* :: $'a$ *set* $\Rightarrow$ $'a$ *cfun* $\Rightarrow$ *bool* **where**
  path-independent-on $A\ f \longleftrightarrow (\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) = f\ (B \cup f\ C))$

abbreviation *path-independent* :: $'a$ *cfun* $\Rightarrow$ *bool* **where**
  path-independent $\equiv$ *path-independent-on UNIV*

Intuitively a choice function satisfying this condition ignores the order in which choices are made in the following sense:

lemma *path-independent-on-symmetric*:
  assumes *f-range-on A f*
  shows *path-independent-on A f* $\longleftrightarrow$ $(\forall B\ C.\ B \subseteq A \wedge C \subseteq A \longrightarrow f\ (B \cup C) = f\ (f\ B \cup f\ C))$

lemma *path-independent-on-Chernoff-on*:
  assumes *path-independent-on A f*
  assumes *f-range-on A f*
  shows *Chernoff-on A f*

lemma *path-independent-on-consistency-on*:
  assumes *path-independent-on A f*
  shows *consistency-on A f*

lemma *Chernoff-on-consistency-on-path-independent-on*:
  assumes *f-range-on A f*
  shows *Chernoff-on A f* $\wedge$ *consistency-on A f* $\longleftrightarrow$ *path-independent-on A f*

lemma (**in** *linear-cf*) *path-independent*:
  shows *path-independent linear-cf*

4.7.1 Path independence and decomposition into orderings

We now show that a choice function over a finite universe satisfying *path-independent* is characterized by taking the maximum elements of some finite set of orderings.

Moulin (1985, Definition 12) says that a choice function is *pseudo-rationalized* by the orderings *Rs* if *f* chooses all of the *greatest* *r* elements of *B* for each $r \in Rs$:

definition *pseudo-rationalizable-on* :: *'a::finite set* $\Rightarrow$ *'a rel set* $\Rightarrow$ *'a cfun* $\Rightarrow$ *bool* **where**
  pseudo-rationalizable-on A Rs f
    $\longleftrightarrow (\forall r{\in}Rs.\ \textit{Linear-order}\ r) \wedge (\forall B{\subseteq}A.\ f\ B = (\bigcup r{\in}Rs.\ \textit{greatest}\ r\ (B \cap \textit{Field}\ r)))$

lemma *pseudo-rationalizable-on-def2*:
  pseudo-rationalizable-on A Rs f
    $\longleftrightarrow (\forall r{\in}Rs.\ \textit{Linear-order}\ r) \wedge (\forall B{\subseteq}A.\ f\ B = (\bigcup r{\in}Rs.\ \textit{set-option}\ (\textit{MaxR.MaxR-opt}\ r\ B)))$

We deviate from Moulin in using non-total linear orders, where his are total, asymmetric, and transitive; in other words, strict total linear orders. This allows us to treat non-decisive choice functions, and we later show that the choice function is decisive iff the orders are total.

Moulin (1985, Theorem 5) assumes *Aizerman* and *Chernoff*, which are equivalent to *path-independent*.

lemma *Aizerman-on-Chernoff-on-path-independent-on*:
  assumes *f-range-on A f*
  shows *Aizerman-on A f* $\wedge$ *Chernoff-on A f* $\longleftrightarrow$ *path-independent-on A f*

It is straightforward to show that pseudo-rationalizable choice functions satisfy *path-independent* using the properties of *MaxR.MaxR-opt*:

lemma *pseudo-rationalizable-on-path-independent-on*:
  assumes *pseudo-rationalizable-on A Rs f*
  shows *path-independent-on A f*

The converse requires that we construct a suitable set of orderings that rationalize $f\ C$ for each $C \subseteq A$. We do this by finding a set $B \subseteq A$ where $f\ B \subseteq C$ by successively removing elements in $f\ A - f\ C$. (As these elements are chosen by f from supersets of B, we rank these above all of those in $f\ B$.) By *consistency* (§4.2), $f\ C = f\ B$. We generate one order for each element of $f\ C$. Some extra care takes care of *decisive* choice functions.

Termination is guaranteed by the finiteness of A and the *f-range-on* hypothesis.

context
  fixes A :: $'a$::*finite set*
  fixes f :: $'a$ *cfun*
  notes *conj-cong*[*fundef-cong*]
begin

function (*domintros*) *mk-linear-orders* :: $'a$ *set* ⇒ $'a$ *set* ⇒ $'a$ *list set* **where**
  mk-linear-orders C B =
   (*if* f $B = \{\}$ *then* $\{[]\}$
    else if f $B \subseteq C$
      then $\{b \# cs \mid b\ cs.\ b \in f\ B \wedge cs \in$ *mk-linear-orders* $\{\}\ (B - \{b\})\}$
      else let b = *SOME* $x.\ x \in f\ B - C$ *in* $\{b \# cs \mid cs.\ cs \in$ *mk-linear-orders* $C\ (B - \{b\})\})$

context
  assumes *f-range-on* A f
begin

lemma *mk-linear-orders-non-empty*:
  assumes $B \subseteq A$
  shows $\exists\, r.\ r \in$ *mk-linear-orders* C B

lemma *mk-linear-orders-range*:
  assumes $r \in$ *mk-linear-orders* C B
  assumes $B \subseteq A$
  shows *set* $r \subseteq B$

lemma *mk-linear-orders-nth*:
  assumes $r \in$ *mk-linear-orders* C B
  assumes $B \subseteq A$
  assumes $i <$ *length* r
  shows r ! $i \in f$ ($B -$ *set* (*take* i r))

lemma *mk-linear-orders-distinct*:
  assumes $r \in$ *mk-linear-orders* C B
  assumes $B \subseteq A$
  shows *distinct* r

lemma *mk-linear-orders-Linear-order*:
  assumes $r \in$ *mk-linear-orders* C A
  shows *Linear-order* (*linord-of-list* r)

lemma *mk-linear-orders-decisive-on-set-r*:
  assumes $r \in$ *mk-linear-orders* C B
  assumes *decisive-on* A f
  assumes $B \subseteq A$
  shows *set* $r = B$

lemma *mk-linear-orders-decisive-on-refl-on*:
  assumes $r \in$ *mk-linear-orders* C A
  assumes *decisive-on* A f
  shows *refl-on* A (*linord-of-list* r)

lemma *mk-linear-orders-decisive-on-total-on*:
  assumes $r \in$ *mk-linear-orders* C A
  assumes *decisive-on* A f
  shows *total-on* A (*linord-of-list* r)

lemma *mk-linear-orders-set-r-decisive-on*:
  assumes $r \in$ *mk-linear-orders* C B
  assumes $B \subseteq A$
  assumes $B \subseteq$ *set* r
  assumes *iia-on* A f
  shows *decisive-on* B f

lemma *mk-linear-orders-total-on-decisive-on*:
  assumes $r \in$ *mk-linear-orders* C A
  assumes $A \subseteq$ *set* r
  assumes *iia-on* A f
  shows *decisive-on* A f

lemma *mk-linear-orders-MaxR-opt-f*:
  assumes $r \in$ *mk-linear-orders* C A
  assumes *MaxR.MaxR-opt* (*linord-of-list* r) D = *Some* x
  assumes *iia-on* A f
  assumes $D \subseteq A$
  shows $x \in f\ D$

lemma *mk-linear-orders-f-MaxR-opt*:
  assumes $x \in f\ C$
  assumes *consistency-on* A f
  assumes $B \subseteq A$
  assumes $C \subseteq B$
  shows $\exists\, r \in$ *mk-linear-orders* C B. *MaxR.MaxR-opt* (*linord-of-list* r) C = *Some* x

end

end

lemma *path-independent-on-pseudo-rationalizable-on*:
  fixes f :: *'a::finite cfun*
  assumes *path-independent-on* A f
  assumes *f-range-on* A f
  assumes *Rs-def*[*simp*]: $Rs = (\bigcup C \in Pow\ A.$ *linord-of-list* ‘ *mk-linear-orders* f C A)
  shows *pseudo-rationalizable-on* A Rs f $\wedge$ ($\forall\, r \in Rs$. *refl-on* A r $\wedge$ *total-on* A r $\longleftrightarrow$ *decisive-on* A f)

Our top-level theorem is essentially Moulin (1985, Theorem 5):

theorem *pseudo-rationalizable*:
  assumes *f-range-on* A f
  shows *path-independent-on* A f
    $\longleftrightarrow$ ($\exists\, Rs$. *pseudo-rationalizable-on* A Rs f $\wedge$ ($\forall\, r \in Rs$. *refl-on* A r $\wedge$ *total-on* A r $\longleftrightarrow$ *decisive-on* A f))

5 Hatfield and Milgrom (2005): Matching with contracts

We take the original paper on matching with contracts by Hatfield and Milgrom (2005) as our roadmap, which follows a similar path to Roth and Sotomayor (1990, §2.5). We defer further motivation to these texts. Our first move is to capture the scenarios described in their §I(A) (p916) in a locale.

locale *Contracts* =
  fixes *Xd* :: *'x::finite* ⇒ *'d::finite*
  fixes *Xh* :: *'x* ⇒ *'h::finite*
  fixes *Pd* :: *'d* ⇒ *'x rel*
  fixes *Ch* :: *'h* ⇒ *'x cfun*

assumes *Pd-linear*: $\forall\, d.\ \mathit{Linear\text{-}order}\ (Pd\ d)$
assumes *Pd-range*: $\forall\, d.\ \mathit{Field}\ (Pd\ d) \subseteq \{x.\ Xd\ x = d\}$
assumes *Ch-range*: $\forall\, h.\ \forall\, X.\ Ch\ h\ X \subseteq \{x{\in}X.\ Xh\ x = h\}$
assumes *Ch-singular*: $\forall\, h.\ \forall\, X.\ \mathit{inj\text{-}on}\ Xd\ (Ch\ h\ X)$
begin

The set of contracts is modelled by the type $'x$, a free type variable that will later be interpreted by some non-empty set. Similarly $'d$ and $'h$ track the names of doctors and hospitals respectively. All of these are finite by virtue of belonging to the *finite* type class.

We fix four constants:

- *Xd* (*Xh*) projects the name of the relevant doctor (hospital) from a contract;
- *Pd* maps doctors to their linear preferences over some subset of contracts that name them (assumptions *Pd-linear* and *Pd-range*); and
- *Ch* maps hospitals to their choice functions (§4), which are similarly constrained to contracts that name them (assumption *Ch-range*). Moreover their choices must name each doctor at most once, i.e., *Xd* must be injective on these (assumption *Ch-singular*).

The reader familiar with the literature will note that we do not have a null contract (also said to represent the *outside option* of unemployment), and instead use partiality of the doctors' preferences. This provides two benefits: firstly, *Xh* is a total function here, and secondly we achieve some economy of description when instantiating this locale as *Pd* only has to rank the relevant contracts.

We note in passing that neither the doctors' nor hospitals' choice functions are required to be decisive, unlike the standard literature on choice functions (§4).

In addition to the above, the following constitute the definitions that must be trusted for the results we prove to be meaningful.

definition $Cd :: {'d} \Rightarrow {'x}\ \mathit{cfun}$ **where**
$Cd\ d \equiv \mathit{set\text{-}option} \circ \mathit{MaxR.MaxR\text{-}opt}\ (Pd\ d)$

definition $\mathit{CD\text{-}on} :: {'d}\ \mathit{set} \Rightarrow {'x}\ \mathit{cfun}$ **where**
$\mathit{CD\text{-}on}\ ds\ X = (\bigcup d{\in}ds.\ Cd\ d\ X)$

abbreviation $CD :: {'x}\ \mathit{set} \Rightarrow {'x}\ \mathit{set}$ **where**
$CD \equiv \mathit{CD\text{-}on}\ \mathit{UNIV}$

definition $CH :: {'x}\ \mathit{cfun}$ **where**
$CH\ X = (\bigcup h.\ Ch\ h\ X)$

The function *Cd* constructs a choice function from the doctor's linear preferences (see §4.6). Both *CD* and *CH* simply aggregate opinions in the obvious way. The functions *CD-on* is parameterized with a set of doctors to support the proofs of §5.5.

We also define *RD* (*Rh*, *RH*) to be the subsets of a given set of contracts that are rejected by the doctors (hospitals). (The abbreviation *Rf* is defined in §4.)

abbreviation (*input*) $\mathit{RD\text{-}on} :: {'d}\ \mathit{set} \Rightarrow {'x}\ \mathit{cfun}$ **where**
$\mathit{RD\text{-}on}\ ds \equiv Rf\ (\mathit{CD\text{-}on}\ ds)$

abbreviation (*input*) $RD :: {'x}\ \mathit{cfun}$ **where**
$RD \equiv \mathit{RD\text{-}on}\ \mathit{UNIV}$

abbreviation (*input*) $Rh :: {'h} \Rightarrow {'x}\ \mathit{cfun}$ **where**
$Rh\ h \equiv Rf\ (Ch\ h)$

abbreviation (*input*) $RH :: {'x}\ \mathit{cfun}$ **where**
$RH \equiv Rf\ CH$

A *mechanism* maps doctor and hospital preferences into a match (here a set of contracts).

type-synonym (in $-$) $({'d}, {'h}, {'x})\ \mathit{mechanism} = ({'d} \Rightarrow {'x}\ \mathit{rel}) \Rightarrow ({'h} \Rightarrow {'x}\ \mathit{cfun}) \Rightarrow {'d}\ \mathit{set} \Rightarrow {'x}\ \mathit{set}$

An *allocation* is a set of contracts where each names a distinct doctor. (Hospitals can contract multiple doctors.)

abbreviation (*input*) *allocation* :: *'x set* ⇒ *bool* **where**
allocation ≡ *inj-on Xd*

We often wish to extract a doctor's or a hospital's contract from an *allocation*.

definition *dX* :: *'x set* ⇒ *'d* ⇒ *'x set* **where**
$dX\ X\ d = \{x \in X.\ Xd\ x = d\}$

definition *hX* :: *'x set* ⇒ *'h* ⇒ *'x set* **where**
$hX\ X\ h = \{x \in X.\ Xh\ x = h\}$

Stability is the key property we look for in a match (here a set of contracts), and consists of two parts.

Firstly, we ask that it be *individually rational*, i.e., the contracts in the match are actually acceptable to all participants. Note that this implies the match is an *allocation*.

definition *individually-rational-on* :: *'d set* ⇒ *'x set* ⇒ *bool* **where**
$\textit{individually-rational-on}\ ds\ X \longleftrightarrow \textit{CD-on}\ ds\ X = X \wedge CH\ X = X$

abbreviation *individually-rational* :: *'x set* ⇒ *bool* **where**
individually-rational ≡ *individually-rational-on UNIV*

The second condition requires that there be no coalition of a hospital and one or more doctors who prefer another set of contracts involving them; the hospital strictly, the doctors weakly. Contrast this definition with the classical one for stable marriages given in §2.

definition *blocking-on* :: *'d set* ⇒ *'x set* ⇒ *'h* ⇒ *'x set* ⇒ *bool* **where**
$\textit{blocking-on}\ ds\ X\ h\ X' \longleftrightarrow X' \neq Ch\ h\ X \wedge X' = Ch\ h\ (X \cup X') \wedge X' \subseteq \textit{CD-on}\ ds\ (X \cup X')$

definition *stable-no-blocking-on* :: *'d set* ⇒ *'x set* ⇒ *bool* **where**
$\textit{stable-no-blocking-on}\ ds\ X \longleftrightarrow (\forall h\ X'.\ \neg \textit{blocking-on}\ ds\ X\ h\ X')$

abbreviation *stable-no-blocking* :: *'x set* ⇒ *bool* **where**
stable-no-blocking ≡ *stable-no-blocking-on UNIV*

definition *stable-on* :: *'d set* ⇒ *'x set* ⇒ *bool* **where**
$\textit{stable-on}\ ds\ X \longleftrightarrow \textit{individually-rational-on}\ ds\ X \wedge \textit{stable-no-blocking-on}\ ds\ X$

abbreviation *stable* :: *'x set* ⇒ *bool* **where**
stable ≡ *stable-on UNIV*

end

5.1 Theorem 1: Existence of stable pairs

We proceed to define a function whose fixed points capture all stable matches. Hatfield and Milgrom (2005, I(B), p917) provide the following intuition:

> The first theorem states that a set of contracts is stable if any alternative contract would be rejected by some doctor or some hospital from its suitably defined opportunity set. In the formulas below, think of the doctors' opportunity set as *XD* and the hospitals' opportunity set as *XH*. If X' is the corresponding stable set, then *XD* must include, in addition to X', all contracts that would not be rejected by the hospitals, and *XH* must similarly include X' and all contracts that would not be rejected by the doctors. If X' is stable, then every alternative contract is rejected by somebody, so $X = XH \cup XD$ [where X is the set of all contracts]. This logic is summarized in the first theorem.

See also Fleiner (2003, p6,§4) and Fleiner (2002, §2), from whom we adopt the term *stable pair*.

context *Contracts*
begin

definition *stable-pair-on* :: *'d set* ⇒ *'x set* × *'x set* ⇒ *bool* **where**

stable-pair-on ds = (λ(*XD*, *XH*). *XD* = − *RH XH* ∧ *XH* = − *RD-on ds XD*)

abbreviation *stable-pair* :: *'x set* × *'x set* ⇒ *bool* **where**
stable-pair ≡ *stable-pair-on UNIV*

abbreviation *match* :: *'x set* × *'x set* ⇒ *'x set* **where**
match X ≡ *fst X* ∩ *snd X*

Hatfield and Milgrom (2005, Theorem 1) state that every solution (*XD*, *XH*) of *stable-pair* yields a stable match *XD* ∩ *XH*, and conversely, i.e., every stable match is the intersection of some stable pair. Aygün and Sönmez (2012b) show that neither is the case without further restrictions on the hospitals' choice functions *Ch*; we exhibit their counterexample below.

Even so we can establish some properties in the present setting:

lemma *stable-pair-on-CD-on*:
assumes *stable-pair-on ds XD-XH*
shows *match XD-XH* = *CD-on ds* (*fst XD-XH*)

lemma *stable-pair-on-CH*:
assumes *stable-pair-on ds XD-XH*
shows *match XD-XH* = *CH* (*snd XD-XH*)

lemma *stable-pair-on-CD-on-CH*:
assumes *stable-pair-on ds XD-XH*
shows *CD-on ds* (*fst XD-XH*) = *CH* (*snd XD-XH*)

lemma *stable-pair-on-allocation*:
assumes *stable-pair-on ds XD-XH*
shows *allocation* (*match XD-XH*)

We run out of steam on the following two lemmas, which are the remaining requirements for stability.

lemma
assumes *stable-pair-on ds XD-XH*
shows *individually-rational-on ds* (*match XD-XH*)
oops

lemma
assumes *stable-pair-on ds XD-XH*
shows *stable-no-blocking* (*match XD-XH*)
oops

Hatfield and Milgrom (2005) also claim that the converse holds:

lemma
assumes *stable-on ds X*
obtains *XD-XH* **where** *stable-pair-on ds XD-XH* **and** *X* = *match XD-XH*
oops

Again, the following counterexample shows that the *substitutes* condition on *Ch* is too weak to guarantee this. We show it holds under stronger assumptions in §5.1.3.

end

5.1.1 Theorem 1 does not hold (Aygün and Sönmez 2012b)

The following counterexample, due to Aygün and Sönmez (2012b, §3: Example 2), comprehensively demonstrates that Hatfield and Milgrom (2005, Theorem 1) does not hold.

We create three types: *D2* consists of two elements, representing the doctors, and *H* is the type of the single hospital. There are four contracts in the type *X4*.

datatype *D2* = *D1* | *D2*
datatype *H1* = *H*
datatype *X4* = *Xd1* | *Xd1'* | *Xd2* | *Xd2'*
primrec *X4d* :: *X4* ⇒ *D2* **where**
  X4d Xd1 = *D1*
| *X4d Xd1'* = *D1*
| *X4d Xd2* = *D2*
| *X4d Xd2'* = *D2*

abbreviation *X4h* :: *X4* ⇒ *H1* **where**
  X4h - ≡ *H*

primrec *PX4d* :: *D2* ⇒ *X4 rel* **where**
  PX4d D1 = *linord-of-list* [*Xd1'*, *Xd1*]
| *PX4d D2* = *linord-of-list* [*Xd2*, *Xd2'*]

function *CX4h* :: *H1* ⇒ *X4 cfun* **where**
  CX4h - {*Xd1*} = {*Xd1*}
| *CX4h* - {*Xd1'*} = {*Xd1'*}
| *CX4h* - {*Xd2*} = {*Xd2*}
| *CX4h* - {*Xd2'*} = {*Xd2'*}
| *CX4h* - {*Xd1*, *Xd1'*} = {*Xd1*}
| *CX4h* - {*Xd1*, *Xd2*} = {*Xd1*, *Xd2*}
| *CX4h* - {*Xd1*, *Xd2'*} = {*Xd2'*}
| *CX4h* - {*Xd1'*, *Xd2*} = {*Xd1'*}
| *CX4h* - {*Xd1'*, *Xd2'*} = {*Xd1'*, *Xd2'*}
| *CX4h* - {*Xd2*, *Xd2'*} = {*Xd2*}
| *CX4h* - {*Xd1*, *Xd1'*, *Xd2*} = {}
| *CX4h* - {*Xd1*, *Xd1'*, *Xd2'*} = {}
| *CX4h* - {*Xd1*, *Xd2*, *Xd2'*} = {}
| *CX4h* - {*Xd1'*, *Xd2*, *Xd2'*} = {}
| *CX4h* - {*Xd1*, *Xd1'*, *Xd2*, *Xd2'*} = {}
| *CX4h* - {} = {}

interpretation *StableNoDecomp*: *Contracts X4d X4h PX4d CX4h*

There are two stable matches in this model.

lemma *stable*:
  shows *StableNoDecomp.stable X* ⟷ *X* = {*Xd1*, *Xd2*} ∨ *X* = {*Xd1'*, *Xd2'*}

However neither arises from a pair *XD*, *XH* that satisfy *StableNoDecomp.stable-pair*:

lemma *StableNoDecomp-XD-XH*:
  shows *StableNoDecomp.stable-pair* (*XD*, *XH*) ⟷ (*XD* = {} ∧ *XH* = {*Xd1*, *Xd1'*, *Xd2*, *Xd2'*})

proposition
  assumes *StableNoDecomp.stable-pair* (*XD*, *XH*)
  shows ¬*StableNoDecomp.stable* (*XD* ∩ *XH*)

Moreover the converse of Theorem 1 does not hold either: the single decomposition that satisfies *StableNoDecomp.stable-pair* (*StableNoDecomp-XD-XH*) does not yield a stable match:

proposition
  assumes *StableNoDecomp.stable X*
  shows ¬(∃ *XD XH*. *StableNoDecomp.stable-pair* (*XD*, *XH*) ∧ *X* = *XD* ∩ *XH*)

So there is not hope for Hatfield and Milgrom (2005, Theorem 1) as it stands. Note that the counterexample satisfies the *substitutes* condition (see §4.1):

lemma
  shows *substitutes* (*CX4h H*)

Therefore while *substitutes* supports the monotonicity argument that underpins their deferred-acceptance algorithm (see §5.2), it is not enough to rescue Theorem 1. One way forward is to constrain the hospitals' choice functions, which we discuss in the next section.

5.1.2 Theorem 1 holds with *independence of rejected contracts*

Aygün and Sönmez (2012b) propose to rectify this issue by requiring hospitals' choices to satisfy *irc* (§4.2). Reassuringly their counterexample fails to satisfy it:

lemma
  shows ¬*irc* (*CX4h H*)

We adopt this hypothesis by extending the *Contracts* locale:

locale *ContractsWithIRC* = *Contracts* +
  assumes *Ch-irc*: ∀ *h*. *irc* (*Ch h*)
begin

This property requires that *Ch* behave, for example, as follows:

lemma *Ch-domain*:
  shows *Ch h* (*A* ∩ {*x*. *Xh x* = *h*}) = *Ch h A*

lemmas *Ch-irc-idem* = *consistency-on-f-idem*[*OF Ch-f-range Ch-consistency*, *simplified*]

lemma *CH-irc-idem*:
  shows *CH* (*CH A*) = *CH A*

lemma *Ch-CH-irc-idem*:
  shows *Ch h* (*CH A*) = *Ch h A*

This suffices to show the left-to-right direction of Theorem 1.

lemma *stable-pair-on-individually-rational*:
  assumes *stable-pair-on ds XD-XH*
  shows *individually-rational-on ds* (*match XD-XH*)

lemma *stable-pair-on-stable-no-blocking-on*:
  assumes *stable-pair-on ds XD-XH*
  shows *stable-no-blocking-on ds* (*match XD-XH*)
proof(*rule stable-no-blocking-onI*)
  fix *h X″*
  assume *C*: *X″* = *Ch h* (*match XD-XH* ∪ *X″*)
  assume *NE*: *X″* ≠ *Ch h* (*match XD-XH*)
  assume *CD*: *X″* ⊆ *CD-on ds* (*match XD-XH* ∪ *X″*)
  have *X″* ⊆ *snd XD-XH*
  proof −
    from *CD* **have** *X″* ⊆ *CD-on ds* (*CD-on ds* (*fst XD-XH*) ∪ *X″*) **by** (*simp only*: *stable-pair-on-CD-on*[*OF assms*])
    then have *X″* ⊆ *CD-on ds* (*fst XD-XH* ∪ *X″*)
      using *CD-on-path-independent* **unfolding** *path-independent-def* **by** (*simp add*: *Un-commute*)
    moreover have *fst XD-XH* ∩ *CD-on ds* (*fst XD-XH* ∪ *X″*) ⊆ *CD-on ds* (*fst XD-XH*)
      using *CD-on-Chernoff* **unfolding** *Chernoff-on-def* **by** (*simp add*: *inf-commute*)

ultimately show *?thesis* **using** *assms* **unfolding** *stable-pair-on-def split-def* **by** *blast*
qed
then have *Ch h* (*snd XD-XH*) = *Ch h* (*Ch h* (*snd XD-XH*) ∪ *X''*)
by (*force intro*!: *consistencyD*[*OF Ch-consistency*] *dest*: *Ch-range'*)
moreover from *NE* **have** *X''* ≠ *Ch h* (*snd XD-XH*)
using *stable-pair-on-CH*[*OF assms*] *CH-domain*[*of - h*] *Ch-domain*[*of h*] **by** (*metis Ch-irc-idem*)
ultimately have *X''* ≠ *Ch h* (*match XD-XH* ∪ *X''*)
using *stable-pair-on-CH*[*OF assms*] *CH-domain*[*of - h*] *Ch-domain*[*of h*]
by (*metis* (*no-types*, *lifting*) *inf.right-idem inf-sup-distrib2*)
with *C* **show** *False* **by** *blast*
qed

theorem *stable-pair-on-stable-on*:
assumes *stable-pair-on ds XD-XH*
shows *stable-on ds* (*match XD-XH*)

end

5.1.3 The converse of Theorem 1

The forward direction of Theorem 1 gives us a way of finding stable matches by computing fixed points of a function closely related to *stable-pair* (see §5.2). The converse says that every stable match can be decomposed in this way, which implies that the stable matches form a lattice (see also §5.2).

The following proofs assume that the hospitals' choice functions satisfy *substitutes* and *irc*.

context *ContractsWithIRC*
begin

context
fixes *ds* :: *'b set*
fixes *X* :: *'a set*
begin

Following Hatfield and Milgrom (2005, Proof of Theorem 1), we partition the set of all contracts into [*X*, *XD-smallest* − *X*, *XH-largest* − *X*] with careful definitions of the two sets *XD-smallest* and *XH-largest*. Specifically *XH-largest* contains all contracts ranked at least as good as those in *X* by the doctors, considering unemployment and unacceptable contracts. Similarly *XD-smallest* contains those ranked at least as poorly.

definition *XH-largest* :: *'a set* **where**
XH-largest =
{*y*. *Xd y* ∈ *ds*
∧ *y* ∈ *Field* (*Pd* (*Xd y*))
∧ (∀ *x* ∈ *dX X* (*Xd y*). (*x*, *y*) ∈ *Pd* (*Xd y*))}

definition *XD-smallest* :: *'a set* **where**
XD-smallest = − (*XH-largest* − *X*)

context
assumes *stable-on ds X*
begin

lemma *Ch-XH-largest-Field*:
assumes *x* ∈ *Ch h XH-largest*
shows *x* ∈ *Field* (*Pd* (*Xd x*))
using *assms* **unfolding** *XH-largest-def* **by** (*blast dest*: *Ch-range'*)

lemma *Ch-XH-largest-Xd*:
assumes *x* ∈ *Ch h XH-largest*
shows *Xd x* ∈ *ds*
using *assms* **unfolding** *XH-largest-def* **by** (*blast dest*: *Ch-range'*)

lemma *X-subseteq-XH-largest*:
  shows $X \subseteq$ *XH-largest*
proof(*rule subsetI*)
  fix x **assume** $x \in X$
  then obtain d **where** $d \in$ *ds* $x \in$ *Cd d X* **using** *stable-on-CD-on*[*OF* ‹*stable-on ds X*›] **unfolding** *CD-on-def* **by** *blast*
  with ‹*stable-on ds X*› **show** $x \in$ *XH-largest*
    using *Pd-linear′ Pd-range′ Cd-range subset-Image1-Image1-iff*[*of Pd d*] *stable-on-allocation*[*of ds X*]
    unfolding *XH-largest-def linear-order-on-def partial-order-on-def stable-on-def inj-on-def dX-def*
    by *simp blast*
qed

lemma *X-subseteq-XD-smallest*:
  shows $X \subseteq$ *XD-smallest*
unfolding *XD-smallest-def* **by** *blast*

lemma *X-XD-smallest-XH-largest*:
  shows $X =$ *XD-smallest* $\cap$ *XH-largest*
using *X-subseteq-XH-largest* **unfolding** *XD-smallest-def* **by** *blast*

The goal of the next few lemmas is to show the constituents of *stable-pair-on ds* (*XD-smallest*, *XH-largest*). Intuitively, if a doctor has a contract x in X, then all of their contracts in *XH-largest* are at least as desirable as x, and so the *stable-no-blocking* hypothesis guarantees the hospitals choose x from *XH-largest*, and similarly the doctors x from *XD-smallest*.

lemma *XH-largestCdXXH-largest*:
  assumes $x \in$ *Ch h XH-largest*
  shows $x \in$ *Cd* (*Xd x*) ($X \cup$ *Ch h XH-largest*)
proof −
  from *assms* **have** $(y, x) \in$ *Pd* (*Xd x*) **if** *Xd y* = *Xd x* **and** $y \in X$ **for** y
    using *that* **by** (*fastforce simp*: *XH-largest-def dX-def dest*: *Ch-range′*)
  with *Ch-XH-largest-Field*[*OF assms*] *Pd-linear Pd-range* **show** *?thesis*
    using *assms Ch-XH-largest-Field*[*OF assms*]
    by (*clarsimp simp*: *Cd-greatest greatest-def*)
       (*metis Ch-singular Pd-range′ inj-onD subset-refl underS-incl-iff*)
qed

lemma *CH-XH-largest*:
  shows *CH XH-largest* = X
proof −
  have *Ch h XH-largest* $\subseteq$ *CD-on ds* ($X \cup$ *Ch h XH-largest*) **for** h
    using *XH-largestCdXXH-largest Ch-XH-largest-Xd Ch-XH-largest-Field* **unfolding** *CD-on-def* **by** *blast*
  from ‹*stable-on ds X*› **have** *Ch h XH-largest* = *Ch h X* **for** h
    using ‹*Ch h XH-largest* $\subseteq$ *CD-on ds* ($X \cup$ *Ch h XH-largest*)› *X-subseteq-XH-largest*
    by − (*erule stable-on-blocking-onD*[**where** *h*=*h* **and** *X″*=*Ch h XH-largest*];
          force intro!: *consistencyD*[*OF Ch-consistency*] *dest*: *Ch-range′*)
  with *stable-on-CH*[*OF* ‹*stable-on ds X*›] **show** *?thesis* **unfolding** *CH-def* **by** *simp*
qed

lemma *Cd-XD-smallest*:
  assumes $d \in$ *ds*
  shows *Cd d* (*XD-smallest* $\cap$ *Field* (*Pd d*)) = *Cd d* ($X \cap$ *Field* (*Pd d*))
proof(*cases* $X \cap$ *Field* (*Pd d*) = {})
  case *True*
  with *Pd-range′ Cd-range′*[**where** *X*=*X*] *stable-on-CD-on*[*OF* ‹*stable-on ds X*›] *mem-CD-on-Cd assms*
  have − *XH-largest* $\cap$ *Field* (*Pd d*) = {}
    unfolding *XH-largest-def dX-def* **by** *auto blast*
  then show *?thesis*

unfolding *XD-smallest-def* **by** (*simp add*: *Int-Un-distrib2*)
next
case *False*
with *Pd-linear*′[*of d*] ‹*stable-on ds X*› *stable-on-CD-on stable-on-allocation assms*
show *?thesis*
unfolding *XD-smallest-def order-on-defs total-on-def*
by (*auto 0 0 simp*: *Int-Un-distrib2 Cd-greatest greatest-def XH-largest-def dX-def*)
(*metis* (*mono-tags*, *lifting*) *IntI Pd-range*′ *UnCI inj-onD*)+
qed

lemma *CD-on-XD-smallest*:
shows *CD-on ds XD-smallest* = *X*
using *stable-on-CD-on*[*OF* ‹*stable-on ds X*›] **unfolding** *CD-on-def2* **by** (*simp add*: *Cd-XD-smallest*)

theorem *stable-on-stable-pair-on*:
shows *stable-pair-on ds* (*XD-smallest*, *XH-largest*)
proof(*rule stable-pair-onI*, *simp-all only*: *prod.sel*)
from *CH-XH-largest* **have** − *RH XH-largest* = − (*XH-largest* − *X*) **by** *blast*
also from *X-XD-smallest-XH-largest* **have** . . . = *XD-smallest* **unfolding** *XD-smallest-def* **by** *blast*
finally show *XD-smallest* = −*RH XH-largest* **by** *blast*
next
from *CD-on-XD-smallest* **have** −*RD-on ds XD-smallest* = −(*XD-smallest* − *X*) **by** *simp*
also have . . . = *XH-largest* **unfolding** *XD-smallest-def* **using** *X-subseteq-XH-largest* **by** *blast*
finally show *XH-largest* = −*RD-on ds XD-smallest* **by** *blast*
qed

end

end

Our ultimate statement of Theorem 1 of Hatfield and Milgrom (2005) ala Aygün and Sönmez (2012b) goes as follows, bearing in mind that we are working in the *ContractsWithIRC* locale:

theorem *T1*:
shows *stable-on ds X* ⟷ (∃ *XD-XH*. *stable-pair-on ds XD-XH* ∧ *X* = *match XD-XH*)
using *stable-pair-on-stable-on stable-on-stable-pair-on X-XD-smallest-XH-largest* **by** *fastforce*

end

5.2 Theorem 3: Algorithmics

Having revived Theorem 1, we reformulate *stable-pair* as a monotone (aka *isotone*) function and exploit the lattice structure of its fixed points, following Hatfield and Milgrom (2005, §II, Theorem 3). This underpins all of their results that we formulate here. Fleiner (2002, §2) provides an intuitive gloss of these definitions.

context *Contracts*
begin

definition *F1* :: ′*x cfun* **where**
F1 X′ = − *RH X*′

definition *F2* :: ′*d set* ⇒ ′*x cfun* **where**
F2 ds X′ = − *RD-on ds X*′

definition *F* :: ′*d set* ⇒ ′*x set* × ′*x set dual* ⇒ ′*x set* × ′*x set dual* **where**
F ds = (λ(*XD*, *XH*). (*F1* (*undual XH*), *dual* (*F2 ds* (*F1* (*undual XH*)))))

We exploit Isabelle/HOL's ordering type classes (over the type constructors ′*a set* and ′*a* × ′*b*) to define *F*. As *F* is *antimono* (where *antimono f* = (∀ *x y*. *x* ≤ *y* ⟶ *f y* ≤ *f x*) for a lattice order ≤) on its second argument *XH*, we adopt the dual lattice order using the type constructor ′*a dual*, where *dual* and *undual* mediate the isomorphism on values, to satisfy Isabelle/HOL's *mono* predicate. Note we work under the *substitutes* hypothesis here. Relating this function to *stable-pair* is syntactically awkward but straightforward:

lemma *fix-F-stable-pair-on*:
  assumes $X = F\ ds\ X$
  shows *stable-pair-on ds* (*map-prod id undual X*)

lemma *stable-pair-on-fix-F*:
  assumes *stable-pair-on ds X*
  shows *map-prod id dual X* = *F ds* (*map-prod id dual X*)

end

The function *F* is monotonic under *substitutes*.

locale *ContractsWithSubstitutes* = *Contracts* +
  assumes *Ch-substitutes*: ∀ *h*. *substitutes* (*Ch h*)
begin

lemma *F1-antimono*:
  shows *antimono F1*

lemma *F2-antimono*:
  shows *antimono* (*F2 ds*)

lemma *F-mono*:
  shows *mono* (*F ds*)

We define the extremal fixed points using Isabelle/HOL's least and greatest fixed point operators:

definition *gfp-F* :: *'b set* ⇒ *'a set* × *'a set* **where**
  gfp-F ds = *map-prod id undual* (*gfp* (*F ds*))

definition *lfp-F* :: *'b set* ⇒ *'a set* × *'a set* **where**
  lfp-F ds = *map-prod id undual* (*lfp* (*F ds*))

lemmas *gfp-F-stable-pair-on* = *fix-F-stable-pair-on*[*OF gfp-unfold*[*OF F-mono*], *folded gfp-F-def*]
lemmas *lfp-F-stable-pair-on* = *fix-F-stable-pair-on*[*OF lfp-unfold*[*OF F-mono*], *folded lfp-F-def*]

These last two lemmas show that the least and greatest fixed points do satisfy *stable-pair*.
Using standard fixed-point properties, we can establish:

lemma *F2-o-F1-mono*:
  shows *mono* (*F2 ds* ∘ *F1*)

lemmas *F2-F1-mono* = *F2-o-F1-mono*[*unfolded o-def*]

lemma *gfp-F-lfp-F*:
  shows *gfp-F ds* = (*F1* (*lfp* (*F2 ds* ∘ *F1*)), *lfp* (*F2 ds* ∘ *F1*))

end

We need hospital CFs to satisfy both *substitutes* and *irc* to relate these fixed points to stable matches.

locale *ContractsWithSubstitutesAndIRC* =
  ContractsWithSubstitutes + *ContractsWithIRC*
begin

lemmas *gfp-F-stable-on* = *stable-pair-on-stable-on*[*OF gfp-F-stable-pair-on*]
lemmas *lfp-F-stable-on* = *stable-pair-on-stable-on*[*OF lfp-F-stable-pair-on*]

end

We demonstrate the effectiveness of our definitions by executing an example due to Hatfield and Milgrom (2005, p920) using Isabelle/HOL's code generator (Haftmann and Nipkow 2010). Note that, while adequate for this toy instance, the representations used here are hopelessly näive: sets are represented by lists and operations typically traverse the entire contract space. It is feasible, with more effort, to derive efficient algorithms; see, for instance, Bijlsma (1991); Bulwahn et al. (2008).

context *ContractsWithSubstitutes*
begin

lemma *gfp-F-code*[*code*]:
  shows *gfp-F ds = map-prod id undual (while (λA. F ds A ≠ A) (F ds) top)*

lemma *lfp-F-code*[*code*]:
  shows *lfp-F ds = map-prod id undual (while (λA. F ds A ≠ A) (F ds) bot)*

end

There are two hospitals and two doctors.

datatype *H2 = H1 | H2*

The contract space is simply the Cartesian product *D2 × H2*.

type-synonym *X-D2-H2 = D2 × H2*

Doctor *D1* prefers *H1 ≻ H2*, doctor *D2* the same *H1 ≻ H2* (but over different contracts).

primrec *P-D2-H2-d :: D2 ⇒ X-D2-H2 rel* **where**
  P-D2-H2-d D1 = linord-of-list [(D1, H1), (D1, H2)]
| *P-D2-H2-d D2 = linord-of-list [(D2, H1), (D2, H2)]*

Hospital *H1* prefers {*D1*} ≻ {*D2*} ≻ ∅, and hospital *H2* {*D1*, *D2*} ≻ {*D1*} ≻ {*D2*} ≻ ∅. We interpret these constraints as follows:

definition *P-D2-H2-H1 :: X-D2-H2 cfun* **where**
  P-D2-H2-H1 A = (if (D1, H1) ∈ A then {(D1, H1)} else if (D2, H1) ∈ A then {(D2, H1)} else {})

definition *P-D2-H2-H2 :: X-D2-H2 cfun* **where**
  P-D2-H2-H2 A =
    (if {(D1, H2), (D2, H2)} ⊆ A then {(D1, H2), (D2, H2)} else
     if (D1, H2) ∈ A then {(D1, H2)} else if (D2, H2) ∈ A then {(D2, H2)} else {})

primrec *P-D2-H2-h :: H2 ⇒ X-D2-H2 cfun* **where**
  P-D2-H2-h H1 = P-D2-H2-H1
| *P-D2-H2-h H2 = P-D2-H2-H2*

Isabelle's code generator requires us to hoist the relevant definitions from the locale to the top-level (see the `codegen` documentation, §7.3).

global-interpretation *P920-example*:
  ContractsWithSubstitutes fst snd P-D2-H2-d P-D2-H2-h
defines *P920-example-gfp-F = P920-example.gfp-F*
    and *P920-example-lfp-F = P920-example.lfp-F*
    and *P920-example-F = P920-example.F*
    and *P920-example-F1 = P920-example.F1*
    and *P920-example-F2 = P920-example.F2*
    and *P920-example-maxR = P920-example.maxR*
    and *P920-example-MaxR-f = P920-example.MaxR-f*
    and *P920-example-Cd = P920-example.Cd*
    and *P920-example-CD-on = P920-example.CD-on*
    and *P920-example-CH = P920-example.CH*

We can now evaluate the *gfp* of *P920-example.F* (i.e., *F* specialized to the above constants) using Isabelle's `value` antiquotation or `eval` method. This yields the (*XD*, *XH*) pair:

$$(\{(D1, H1), (D1, H2), (D2, H2)\}, \{(D1, H1), (D2, H1), (D2, H2)\})$$

The stable match is therefore $\{(D1, H1), (D2, H2)\}$.

The *lfp* of *P920-example.F* is identical to the *gfp*:

$$(\{(D1, H1), (D1, H2), (D2, H2)\}, \{(D1, H1), (D2, H1), (D2, H2)\})$$

This implies that there is only one stable match in this scenario.

5.3 Theorem 4: Optimality

Hatfield and Milgrom (2005, Theorem 4) assert that the greatest fixed point *gfp-F* of *F* yields the stable match most preferred by the doctors in the following sense:

context *Contracts*
begin

definition *doctor-optimal-match* :: $'d$ *set* ⇒ $'x$ *set* ⇒ *bool* **where**
  doctor-optimal-match ds Y
    ⟷ (*stable-on ds Y* ∧ (∀ *X*. ∀ *x*∈*X*. *stable-on ds X* ⟶ (∃ *y* ∈ *Y*. (*x*, *y*) ∈ *Pd* (*Xd x*))))

end

In a similar sense, *lfp-F* is the doctor-pessimal match.

We state a basic doctor-optimality result in terms of *stable-pair* in the *ContractsWithSubstitutes* locale for generality; we can establish *doctor-optimal-match* only under additional constraints on hospital choice functions (see §5.1.2).

context *ContractsWithSubstitutes*
begin

context
  fixes *XD-XH* :: $'a$ *set* × $'a$ *set*
  fixes *ds* :: $'b$ *set*
  assumes *stable-pair-on ds XD-XH*
begin

lemma *gfp-F-upperbound*:
  shows (*fst XD-XH*, *dual* (*snd XD-XH*)) ≤ *gfp* (*F ds*)

lemma *XD-XH-gfp-F*:
  shows *fst XD-XH* ⊆ *fst* (*gfp-F ds*)
    and *snd* (*gfp-F ds*) ⊆ *snd XD-XH*

lemma *lfp-F-upperbound*:
  shows *lfp* (*F ds*) ≤ (*fst XD-XH*, *dual* (*snd XD-XH*))

lemma *XD-XH-lfp-F*:
  shows *fst* (*lfp-F ds*) ⊆ *fst XD-XH*
    and *snd XD-XH* ⊆ *snd* (*lfp-F ds*)

We appeal to the doctors' linear preferences to show the optimality (pessimality) of *gfp-F* (*lfp-F*) for doctors.

theorem *gfp-f-doctor-optimal*:
  assumes *x* ∈ *match XD-XH*

shows $\exists\, y \in$ *match* (*gfp-F ds*). $(x, y) \in$ *Pd* (*Xd x*)

theorem *lfp-f-doctor-pessimal*:
assumes $x \in$ *match* (*lfp-F ds*)
shows $\exists\, y \in$ *match XD-XH*. $(x, y) \in$ *Pd* (*Xd x*)

end

end

theorem (**in** *ContractsWithSubstitutesAndIRC*) *gfp-F-doctor-optimal-match*:
shows *doctor-optimal-match ds* (*match* (*gfp-F ds*))

Conversely *lfp-F* is most preferred by the hospitals in a revealed-preference sense, and *gfp-F* least preferred. These results depend on *Ch-domain* and hence the *irc* hypothesis on hospital choice functions.

context *ContractsWithSubstitutesAndIRC*
begin

theorem *lfp-f-hospital-optimal*:
assumes *stable-pair-on ds XD-XH*
assumes $x \in$ *Ch h* (*match* (*lfp-F ds*))
shows $x \in$ *Ch h* (*match* (*lfp-F ds*) $\cup$ *match XD-XH*)

theorem *gfp-f-hospital-pessimal*:
assumes *stable-pair-on ds XD-XH*
assumes $x \in$ *Ch h* (*match XD-XH*)
shows $x \in$ *Ch h* (*match* (*gfp-F ds*) $\cup$ *match XD-XH*)

end

The general lattice-theoretic results of e.g. Fleiner (2002) depend on the full Tarski-Knaster fixed point theorem, which is difficult to state in the present type class-based setting. (The theorem itself is available in the Isabelle/HOL distribution but requires working with less convenient machinery.)

5.4 Theorem 5 does not hold (Hatfield and Kojima 2008)

Hatfield and Milgrom (2005, Theorem 5) claim that:

> Suppose *H* contains at least two hospitals, which we denote by *h* and *h′*. Further suppose that *Rf* (*Ch h*) is not isotone, that is, contracts are not *substitutes* for *h*. Then there exist preference orderings for the doctors in set *D*, a preference ordering for a hospital *h′* with a single job opening such that, regardless of the preferences of the other hospitals, no stable set of contracts exists.

Hatfield and Kojima (2008, Observation 1) show this is not true: there can be stable matches even if hospital choice functions violate *substitutes*. This motivates looking for conditions weaker than *substitutes* that still guarantee stable matches, a project taken up by Hatfield and Kojima (2010); see §6. We omit their counterexample to this incorrect claim.

5.5 Theorem 6: "Vacancy chain" dynamics

Hatfield and Milgrom (2005, II(C), p923) propose a model for updating a stable match *X* when a doctor d' retires. Intuitively the contracts mentioning d' are discarded and a modified algorithm run from the *XH-largest* and *XD-smallest* sets determined from *X*. The result is another stable match where the remaining doctors $ds - \{d'\}$ are (weakly) better off and the hospitals (weakly) worse off than they were in the initial state. The proofs are essentially the same as for optimality (§5.3).

context *ContractsWithSubstitutesAndIRC*
begin

context
  fixes X :: $'a\ set$
  fixes d' :: $'b$
  fixes ds :: $'b\ set$
begin

Hatfield and Milgrom do not motivate why the process uses this functional and not F.

definition F' :: $'a\ set \times 'a\ set\ dual \Rightarrow 'a\ set \times 'a\ set\ dual$ **where**
  $F' = (\lambda(XD, XH).\ (-\ RH\ (undual\ XH),\ dual\ (-\ RD\text{-}on\ (ds-\{d'\})\ XD)))$

lemma F'*-apply*:
  $F'\ (XD, XH) = (-\ RH\ (undual\ XH),\ dual\ (-\ RD\text{-}on\ (ds - \{d'\})\ XD))$
  by (*simp add*: F'*-def*)

lemma F'*-mono*:
  shows *mono* F'

lemma *fix-*F'*-stable-pair-on*:
  $stable\text{-}pair\text{-}on\ (ds - \{d'\})\ (map\text{-}prod\ id\ undual\ A)$
  if $A = F'\ A$

We model their update process using the *while* combinator, as we cannot connect it to the extremal fixed points as we did in §5.2 because we begin computing from the stable match X.

definition F'*-iter* :: $'a\ set \times 'a\ set\ dual$ **where**
  $F'\text{-}iter = (while\ (\lambda A.\ F'\ A \neq A)\ F'\ (XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X)))$

abbreviation F'*-iter-match* :: $'a\ set$ **where**
  $F'\text{-}iter\text{-}match \equiv match\ (map\text{-}prod\ id\ undual\ F'\text{-}iter)$

context
  assumes *stable-on ds X*
begin

lemma *F-start*:
  shows $F\ ds\ (XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X)) = (XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X))$

lemma F'*-start*:
  shows $(XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X)) \leq F'\ (XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X))$

lemma
  shows F'*-iter-stable-pair-on*: $stable\text{-}pair\text{-}on\ (ds-\{d'\})\ (map\text{-}prod\ id\ undual\ F'\text{-}iter)$ (**is** *?thesis1*)
    and F'*-start-le-*F'*-iter*: $(XD\text{-}smallest\ ds\ X,\ dual\ (XH\text{-}largest\ ds\ X)) \leq F'\text{-}iter$ (**is** *?thesis2*)

lemma F'*-iter-match-stable-on*:
  shows $stable\text{-}on\ (ds-\{d'\})\ F'\text{-}iter\text{-}match$

theorem F'*-iter-match-doctors-weakly-better-off*:
  assumes $x \in Cd\ d\ X$
  assumes $d \neq d'$
  shows $\exists\, y \in Cd\ d\ F'\text{-}iter\text{-}match.\ (x, y) \in Pd\ d$

theorem F'*-iter-match-hospitals-weakly-worse-off*:
  assumes $x \in Ch\ h\ X$
  shows $x \in Ch\ h\ (F'\text{-}iter\text{-}match \cup X)$

Hatfield and Milgrom observe but do not prove that F'*-iter-match* is not necessarily doctor-optimal wrt the new set of doctors, even if X was.

These results seem incomplete. One might expect that the process of reacting to a doctor's retirement would involve considering new entrants to the workforce and allowing the set of possible contracts to be refined. There are also the questions of hospitals opening and closing.

end

end

end

5.6 Theorems 8 and 9: A "rural hospitals" theorem

Given that some hospitals are less desirable than others, the question arises of whether there is a mechanism that can redistribute doctors to under-resourced hospitals while retaining the stability of the match. Roth's *rural hospitals theorem* (Roth and Sotomayor 1990, Theorem 5.12) resolves this in the negative by showing that each doctor and hospital signs the same number of contracts in every stable match. In the context of contracts the theorem relies on the further hypothesis that hospital choices satisfy the law of aggregate demand (§4.3).

locale *ContractsWithLAD* = *Contracts* +
  assumes *Ch-lad*: ∀ *h*. *lad* (*Ch h*)

locale *ContractsWithSubstitutesAndLAD* =
  ContractsWithSubstitutes + *ContractsWithLAD*

We can use results that hold under *irc* by discharging that hypothesis against *lad* using the *lad-on-substitutes-on-irc-on* lemma. This is the effect of the following *sublocale* command:

sublocale *ContractsWithSubstitutesAndLAD* < *ContractsWithSubstitutesAndIRC*
using *Ch-range Ch-substitutes Ch-lad* **by** *unfold-locales* (*blast intro*: *lad-on-substitutes-on-irc-on f-range-onI*)

context *ContractsWithSubstitutesAndLAD*
begin

The following results lead to Hatfield and Milgrom (2005, Theorem 8), and the proofs go as they say. Again we state these with respect to an arbitrary solution to *stable-pair*.

context
  fixes *XD-XH* :: *'a set* × *'a set*
  fixes *ds* :: *'b set*
  assumes *stable-pair-on ds XD-XH*
begin

lemma *Cd-XD-gfp-F-card*:
  assumes *d* ∈ *ds*
  shows *card* (*Cd d* (*fst XD-XH*)) ≤ *card* (*Cd d* (*fst* (*gfp-F ds*)))

lemma *Ch-gfp-F-XH-card*:
  shows *card* (*Ch h* (*snd* (*gfp-F ds*))) ≤ *card* (*Ch h* (*snd XD-XH*))

theorem *Theorem-8*:
  shows *d* ∈ *ds* ⟹ *card* (*Cd d* (*fst XD-XH*)) = *card* (*Cd d* (*fst* (*gfp-F ds*)))
    and *card* (*Ch h* (*snd XD-XH*)) = *card* (*Ch h* (*snd* (*gfp-F ds*)))

end

Their result may be more easily understood when phrased in terms of arbitrary stable matches:

corollary *rural-hospitals-theorem*:
  assumes *stable-on ds X*
  assumes *stable-on ds Y*
  shows *d* ∈ *ds* ⟹ *card* (*Cd d X*) = *card* (*Cd d Y*)

and *card* (*Ch h X*) = *card* (*Ch h Y*)

end

Hatfield and Milgrom (2005, Theorem 9) show that without *lad*, the rural hospitals theorem does not hold. Their proof does not seem to justify the theorem as stated (for instance, the contracts x', y' and z' need not exist), and so we instead simply provide a counterexample (discovered by `nitpick`) to the same effect.

lemma (**in** *ContractsWithSubstitutesAndIRC*) *Theorem-9-counterexample*:
assumes *stable-on ds Y*
assumes *stable-on ds Z*
shows *card* (*Ch h Y*) = *card* (*Ch h Z*)
oops

datatype *X3* = *Xd1* | *Xd1'* | *Xd2*
primrec *X3d* :: *X3* ⇒ *D2* **where**
X3d Xd1 = *D1*
| *X3d Xd1'* = *D1*
| *X3d Xd2* = *D2*

abbreviation *X3h* :: *X3* ⇒ *H1* **where**
X3h - ≡ *H*

primrec *PX3d* :: *D2* ⇒ *X3 rel* **where**
PX3d D1 = *linord-of-list* [*Xd1*, *Xd1'*]
| *PX3d D2* = *linord-of-list* [*Xd2*]

function *CX3h* :: *H1* ⇒ *X3 set* ⇒ *X3 set* **where**
CX3h - {*Xd1*} = {*Xd1*}
| *CX3h* - {*Xd1'*} = {*Xd1'*}
| *CX3h* - {*Xd2*} = {*Xd2*}
| *CX3h* - {*Xd1*, *Xd1'*} = {*Xd1'*}
| *CX3h* - {*Xd1*, *Xd2*} = {*Xd1*, *Xd2*}
| *CX3h* - {*Xd1'*, *Xd2*} = {*Xd1'*}
| *CX3h* - {*Xd1*, *Xd1'*, *Xd2*} = {*Xd1'*}
| *CX3h* - {} = {}

interpretation *Theorem-9*: *ContractsWithSubstitutesAndIRC X3d X3h PX3d CX3h*

lemma *Theorem-9-stable-Xd1'*:
shows *Theorem-9.stable-on UNIV* {*Xd1'*}

lemma *Theorem-9-stable-Xd1-Xd2*:
shows *Theorem-9.stable-on UNIV* {*Xd1*, *Xd2*}

This violates the rural hospitals theorem:

theorem
shows *card* (*Theorem-9.CH* {*Xd1'*}) ≠ *card* (*Theorem-9.CH* {*Xd1*, *Xd2*})

. . . which is attributed to the failure of the hospitals' choice functions to satisfy *lad*:

lemma *CX3h-not-lad*:
shows ¬*lad* (*CX3h h*)

Ciupan et al. (2016) discuss an alternative approach to this result in a marriage market.

5.7 Theorems 15 and 16: Cumulative Offer Processes

The goal of Hatfield and Milgrom (2005, §V) is to connect this theory of contracts with matching to earlier work on auctions by the first of the authors, in particular by eliminating the *substitutes* hypothesis. They do so by defining a *cumulative offer process* (COP):

context *Contracts*
begin

definition *cop-F-HM* :: *'d set* ⇒ *'x set* × *'x set* ⇒ *'x set* × *'x set* **where**
  cop-F-HM ds = (λ(*XD*, *XH*). (− *RH XH*, *XH* ∪ *CD-on ds* (− *RH XH*)))

Intuitively all of the doctors simultaneously offer their most preferred contracts that have yet to be rejected by the hospitals, and the hospitals choose amongst these and all that have been offered previously. Asking hospital choice functions to satisfy the *substitutes* condition effectively forces hospitals to consider only the contracts they have previously not rejected.

This definition is neither monotonic nor increasing (i.e., it is not the case that $\forall x.\ x \le$ *cop-F-HM ds x*). We rectify this by focusing on the second part of the definition.

definition *cop-F* :: *'d set* ⇒ *'x set* ⇒ *'x set* **where**
  cop-F ds XH = *XH* ∪ *CD-on ds* (− *RH XH*)

lemma *cop-F-HM-cop-F*:
  shows *cop-F-HM ds XD-XH* = (− *RH* (*snd XD-XH*), *cop-F ds* (*snd XD-XH*))
unfolding *cop-F-HM-def cop-F-def split-def* **by** *simp*

lemma *cop-F-increasing*:
  shows *x* ≤ *cop-F ds x*

We have the following straightforward case distinction principles:

lemma *cop-F-cases*:
  assumes *x* ∈ *cop-F ds fp*
  obtains (*fp*) *x* ∈ *fp* | (*CD-on*) *x* ∈ *CD-on ds* (−*RH fp*) − *fp*
using *assms* **unfolding** *cop-F-def* **by** *blast*

lemma *CH-cop-F-cases*:
  assumes *x* ∈ *CH* (*cop-F ds fp*)
  obtains (*CH*) *x* ∈ *CH fp* | (*RH-fp*) *x* ∈ *RH fp* | (*CD-on*) *x* ∈ *CD-on ds* (−*RH fp*) − *fp*
using *assms CH-range cop-F-def* **by** *auto*

The existence of fixed points for our earlier definitions (§5.2) was guaranteed by the Tarski-Knaster theorem, which relies on the monotonicity of the defining functional. As *cop-F* lacks this property, we appeal instead to the Bourbaki-Witt theorem for increasing functions.

interpretation *COP*: *bourbaki-witt-fixpoint Sup* {(*x*, *y*). *x* ≤ *y*} *cop-F ds* **for** *ds*

definition *fp-cop-F* :: *'d set* ⇒ *'x set* **where**
  fp-cop-F ds = *COP.fixp-above ds* {}

abbreviation *cop ds* ≡ *CH* (*fp-cop-F ds*)

Given that the set of contracts is finite, we avoid continuity and admissibility issues; we have the following straightforward induction principle:

lemma *fp-cop-F-induct*[*case-names base step*]:
  assumes *P* {}
  assumes ⋀*fp*. *P fp* ⟹ *P* (*cop-F ds fp*)
  shows *P* (*fp-cop-F ds*)

An alternative is to use the *while* combinator, which is equivalent to the above by *COP.fixp-above-conv-while*.

In any case, invariant reasoning is essential to verifying the properties of the COP, no matter how we phrase it. We develop a small program logic to ease the reuse of the invariants we prove.

definition
  $valid :: 'd\ set \Rightarrow ('d\ set \Rightarrow 'x\ set \Rightarrow bool) \Rightarrow ('d\ set \Rightarrow 'x\ set \Rightarrow bool) \Rightarrow bool$
where
  $valid\ ds\ P\ Q = (Q\ ds\ \{\} \wedge (\forall fp.\ P\ ds\ fp \wedge Q\ ds\ fp \longrightarrow Q\ ds\ (cop\text{-}F\ ds\ fp)))$

abbreviation
  $invariant :: 'd\ set \Rightarrow ('d\ set \Rightarrow 'x\ set \Rightarrow bool) \Rightarrow bool$
where
  $invariant\ ds\ P \equiv valid\ ds\ (\lambda\text{-}\ \text{-}.\ True)\ P$

Intuitively *valid ds P Q* asserts that the COP satisfies Q assuming that it satisfies P. This allows us to decompose our invariant proofs. By setting the precondition to *True*, *invariant ds P* captures the proof obligations of *fp-cop-F-induct* exactly.

The following lemmas ease the syntactic manipulation of these facts.

lemma *validI*[*case-names base step*]:
  assumes $Q\ ds\ \{\}$
  assumes $\bigwedge fp.\ [\![P\ ds\ fp;\ Q\ ds\ fp]\!] \Longrightarrow Q\ ds\ (cop\text{-}F\ ds\ fp)$
  shows $valid\ ds\ P\ Q$

lemma *invariant-cop-FD*:
  assumes $invariant\ ds\ P$
  assumes $P\ ds\ fp$
  shows $P\ ds\ (cop\text{-}F\ ds\ fp)$

lemma *invariantD*:
  assumes $invariant\ ds\ P$
  shows $P\ ds\ (fp\text{-}cop\text{-}F\ ds)$

lemma *valid-pre*:
  assumes $valid\ ds\ P'\ Q$
  assumes $\bigwedge fp.\ P\ ds\ fp \Longrightarrow P'\ ds\ fp$
  shows $valid\ ds\ P\ Q$

lemma *valid-invariant*:
  assumes $valid\ ds\ P\ Q$
  assumes $invariant\ ds\ P$
  shows $invariant\ ds\ (\lambda\ ds\ fp.\ P\ ds\ fp \wedge Q\ ds\ fp)$

lemma *valid-conj*:
  assumes $valid\ ds\ (\lambda ds\ fp.\ R\ ds\ fp \wedge P\ ds\ fp \wedge Q\ ds\ fp)\ P$
  assumes $valid\ ds\ (\lambda ds\ fp.\ R\ ds\ fp \wedge P\ ds\ fp \wedge Q\ ds\ fp)\ Q$
  shows $valid\ ds\ R\ (\lambda\ ds\ fp.\ P\ ds\ fp \wedge Q\ ds\ fp)$

end

Hatfield and Milgrom (2005, Theorem 15) assert that *fp-cop-F* is equivalent to the doctor-offering algorithm *gfp-F*, assuming *substitutes*. (Note that the fixed points generated by increasing functions do not necessarily form a lattice, so there is not necessarily a hospital-optimal match, and indeed in general these do not exist.) Our proof is eased by the decomposition lemma *gfp-F-lfp-F* and the standard properties of fixed points in a lattice.

context *ContractsWithSubstitutes*
begin

lemma *lfp-F2-o-F1-fp-cop-F*:
  shows $lfp\ (F2\ ds \circ F1) = fp\text{-}cop\text{-}F\ ds$
proof(*rule antisym*)

```
  have (F2 ds ∘ F1) (fp-cop-F ds) ⊆ cop-F ds (fp-cop-F ds)
    by (clarsimp simp: F2-def F1-def cop-F-def)
  then show lfp (F2 ds ∘ F1) ⊆ fp-cop-F ds
    by (simp add: lfp-lowerbound fp-cop-F-unfold[symmetric])
next
  show fp-cop-F ds ⊆ lfp (F2 ds ∘ F1)
  proof(induct rule: fp-cop-F-induct)
    case base then show ?case by simp
  next
    case (step fp) note IH = ‹fp ⊆ lfp (F2 ds ∘ F1)›
    then have CD-on ds (− RH fp) ⊆ lfp (F2 ds ∘ F1)
      apply (subst lfp-unfold[OF F2-o-F1-mono])
      by (smt (verit, ccfv-SIG) Compl-iff Contracts.F1-def Contracts.F2-def Contracts-axioms DiffD2
          F2-o-F1-mono comp-eq-dest-lhs in-mono monoD subsetI)
    with IH show ?case
      unfolding cop-F-def by blast
  qed
qed

theorem Theorem-15:
  shows gfp-F ds = (− RH (fp-cop-F ds), fp-cop-F ds)
using lfp-F2-o-F1-fp-cop-F unfolding gfp-F-lfp-F F1-def by simp

theorem Theorem-15-match:
  shows match (gfp-F ds) = CH (fp-cop-F ds)
using Theorem-15 by (fastforce dest: subsetD[OF CH-range])

end
```

With some auxiliary definitions, we can evaluate the COP on the example from §5.2.

```
lemma P920-example-fp-cop-F-value:
  shows P920-example-CH (P920-example-fp-cop-F UNIV) = {(D1, H1), (D2, H2)}
by eval
```

Hatfield and Milgrom (2005, Theorem 16) assert that this process yields a stable match when we have a single hospital (now called an auctioneer) with unrestricted preferences. As before, this holds provided the auctioneer's preferences satisfy *irc*.

We begin by establishing two obvious invariants of the COP that hold in general.

```
context Contracts
begin

definition cop-F-range-inv :: 'd set ⇒ 'x set ⇒ bool where
  cop-F-range-inv ds fp ⟷ (∀x∈fp. x ∈ Field (Pd (Xd x)) ∧ Xd x ∈ ds)

definition cop-F-closed-inv :: 'd set ⇒ 'x set ⇒ bool where
  cop-F-closed-inv ds fp ⟷ (∀x∈fp. above (Pd (Xd x)) x ⊆ fp)
```

The first, *cop-F-range-inv*, simply states that the result of the COP respects the structural conditions for doctors. The second *cop-F-closed-inv* states that the COP is upwards-closed with respect to the doctors' preferences.

```
lemma cop-F-range-inv:
  shows invariant ds cop-F-range-inv
unfolding valid-def cop-F-range-inv-def cop-F-def by (fastforce simp: mem-CD-on-Cd dest: Cd-range')

lemma cop-F-closed-inv:
  shows invariant ds cop-F-closed-inv
unfolding valid-def cop-F-closed-inv-def cop-F-def above-def
by (clarsimp simp: subset-iff) (metis Cd-preferred ComplI Un-upper1 mem-CD-on-Cd subsetCE)
```

lemmas *fp-cop-F-range-inv* = *invariantD*[*OF cop-F-range-inv*]
lemmas *fp-cop-F-range-inv′* = *fp-cop-F-range-inv*[*unfolded cop-F-range-inv-def*, *rule-format*]
lemmas *fp-cop-F-closed-inv* = *invariantD*[*OF cop-F-closed-inv*]
lemmas *fp-cop-F-closed-inv′* = *subsetD*[*OF bspec*[*OF invariantD*[*OF cop-F-closed-inv*, *unfolded cop-F-closed-inv-def*, *simplified*]]]

The only challenge in showing that the COP yields a stable match is in establishing *stable-no-blocking-on*. Our key lemma states that that if *CH* rejects all contracts for doctor *d* in *fp-cop-F*, then all contracts for *d* are in *fp-cop-F*.

lemma *cop-F-RH*:
  assumes $d \in ds$
  assumes $x \in$ *Field* (*Pd d*)
  assumes *aboveS* (*Pd d*) $x \subseteq$ *RH fp*
  shows $x \in$ *cop-F ds fp*

lemma *fp-cop-F-all*:
  assumes $d \in ds$
  assumes $d \notin$ *Xd* ‘ *CH* (*fp-cop-F ds*)
  shows *Field* (*Pd d*) $\subseteq$ *fp-cop-F ds*

Aygün and Sönmez (2012b) observe that any blocking contract must be weakly preferred by its doctor to anything in the outcome of the *fp-cop-F*:

lemma *fp-cop-F-preferred*:
  assumes $y \in$ *CD-on ds* (*CH* (*fp-cop-F ds*) ∪ X'')
  assumes $x \in$ *CH* (*fp-cop-F ds*)
  assumes *Xd x* = *Xd y*
  shows $(x, y) \in$ *Pd* (*Xd x*)

The headline lemma cobbles these results together.

lemma X''*-closed*:
  assumes $X'' \subseteq$ *CD-on ds* (*CH* (*fp-cop-F ds*) ∪ X'')
  shows $X'' \subseteq$ *fp-cop-F ds*
proof(*rule subsetI*)
  fix *x* **assume** $x \in X''$
  show $x \in$ *fp-cop-F ds*
  proof(*cases Xd x* ∈ *Xd* ‘ *CH* (*fp-cop-F ds*))
    case *True*
    then obtain *y* **where** *Xd y* = *Xd x* **and** $y \in$ *CH* (*fp-cop-F ds*) **by** *clarsimp*
    with *assms* ‹$x \in X''$› **show** *?thesis*
      using *CH-range fp-cop-F-closed-inv′ fp-cop-F-preferred* **unfolding** *above-def* **by** *blast*
  next
    case *False* **with** *assms* ‹$x \in X''$› **show** *?thesis*
      by (*meson Cd-range′ IntD2 fp-cop-F-all mem-CD-on-Cd rev-subsetD*)
  qed
qed

The *irc* constraint on the auctioneer’s preferences is needed for *stable-no-blocking* and their part of *individually-rational*.

end

context *ContractsWithIRC*
begin

lemma *cop-stable-no-blocking-on*:
  shows *stable-no-blocking-on ds* (*cop ds*)
proof(*rule stable-no-blocking-onI*)

```
  fix h X''
  assume C: X'' = Ch h (CH (fp-cop-F ds) ∪ X'')
  assume NE: X'' ≠ Ch h (CH (fp-cop-F ds))
  assume CD: X'' ⊆ CD-on ds (CH (fp-cop-F ds) ∪ X'')
  from CD have X'' ⊆ fp-cop-F ds by (rule X''-closed)
  then have X: CH (fp-cop-F ds) ∪ X'' ⊆ fp-cop-F ds using CH-range by simp
  from C NE Ch-CH-irc-idem[of h]  show False
    using consistency-onD[OF Ch-consistency - X] CH-domain Ch-domain by blast
qed

theorem Theorem-16:
  assumes h: (UNIV::'c set) = {h}
  shows stable-on ds (cop ds) (is stable-on ds ?fp)
proof(rule stable-onI)
  show individually-rational-on ds ?fp
  proof(rule individually-rational-onI)
    from h have allocation ?fp by (simp add: Ch-singular CH-Ch-singular)
    then show CD-on ds ?fp = ?fp
      by (rule CD-on-closed) (blast dest: CH-range' fp-cop-F-range-inv')
    show CH (CH (fp-cop-F ds)) = CH (fp-cop-F ds) by (simp add: CH-irc-idem)
  qed
  show stable-no-blocking-on ds ?fp by (rule cop-stable-no-blocking-on)
qed

end
```

5.8 Concluding remarks

From Hatfield and Milgrom (2005), we have not shown Theorems 2, 7, 13 and 14, all of which are intended to position their results against prior work in this space. We delay establishing their strategic results (Theorems 10, 11 and 12) to §8, after we have developed more useful invariants for the COP.

By assuming *irc*, Aygün and Sönmez (2012b) are essentially trading on Plott's path independence condition (§4.7), as observed by Chambers and Yenmez (2013). The latter show that these results generalize naturally to many-to-many matches, where doctors also use path-independent choice functions; see also Fleiner (2003).

For many applications, however, *substitutes* proves to be too strong a condition. The COP of §5.7 provides a way forward, as we discuss in the next section.

6 Hatfield and Kojima (2010): Substitutes and stability for matching with contracts

Hatfield and Kojima (2010) set about weakening *substitutes* and therefore making the *cumulative offer processes* (COPs, §5.7) applicable to more matching problems. In doing so they lose the lattice structure of the stable matches, which necessitates redeveloping the results of §5.

In contrast to the COP of §5.7, Hatfield and Kojima (2010) develop and analyze a *single-offer* variant, where only one doctor (who has no held contract) proposes per round. The order of doctors making offers is not specified. We persist with the simultaneous-offer COP as it is deterministic. See Hirata and Kasuya (2014) for equivalence arguments.

We begin with some observations due to Aygün and Sönmez. Firstly, as for the matching-with-contracts setting of §5, Aygün and Sönmez (2012a) demonstrate that these results depend on hospital preferences satisfying *irc*. We do not formalize their examples. Secondly, an alternative to hospitals having choice functions (as we have up to now) is for the hospitals to have preference orders over sets, which is suggested by both Hatfield and Milgrom (2005) (weakly) and Hatfield and Kojima (2010). Aygün and Sönmez (2012a, §2) argue that this approach is under-specified and propose to define *Ch* as choosing amongst maximal elements of some non-strict preference order (i.e., including indifference). They then claim that this is equivalent to taking *Ch* as primitive, and so we continue down that path.

6.1 Theorem 1: the COP yields a stable match under *bilateral substitutes*

The weakest replacement condition suggested by Hatfield and Kojima (2010, §1) for the *substitutes* condition on hospital choice functions is termed *bilateral substitutes*:

> Contracts are *bilateral substitutes* for a hospital if there are no two contracts x and z and a set of contracts Y with other doctors than those associated with x and z such that the hospital that regards Y as available wants to sign z if and only if x becomes available. In other words, contracts are bilateral substitutes when any hospital, presented with an offer from a doctor he does not currently employ, never wishes to also hire another doctor he does not currently employ at a contract he previously rejected.

Note that this constraint is specific to this matching-with-contracts setting, unlike those of §4.

context *Contracts*
begin

definition *bilateral-substitutes-on* :: $'x\ set \Rightarrow 'x\ cfun \Rightarrow bool$ **where**
bilateral-substitutes-on A f
$\longleftrightarrow \neg(\exists\, B {\subseteq} A.\ \exists\, a\ b.\ \{a,\ b\} \subseteq A \wedge Xd\ a \notin Xd\ \text{‘}\ B \wedge Xd\ b \notin Xd\ \text{‘}\ B$
$\wedge\ b \notin f\ (B \cup \{b\}) \wedge b \in f\ (B \cup \{a,\ b\}))$

abbreviation *bilateral-substitutes* :: $'x\ cfun \Rightarrow bool$ **where**
bilateral-substitutes $\equiv$ *bilateral-substitutes-on UNIV*

lemma *bilateral-substitutes-on-def2*:
bilateral-substitutes-on A f
$\longleftrightarrow (\forall\, B {\subseteq} A.\ \forall\, a {\in} A.\ \forall\, b {\in} A.\ Xd\ a \notin Xd\ \text{‘}\ B \wedge Xd\ b \notin Xd\ \text{‘}\ B \wedge b \notin f\ (B \cup \{b\}) \longrightarrow b \notin f\ (B \cup \{a,\ b\}))$

lemma *substitutes-on-bilateral-substitutes-on*:
assumes *substitutes-on A f*
shows *bilateral-substitutes-on A f*

Aygün and Sönmez (2012a, §4, Definition 5) give the following equivalent definition:

lemma *bilateral-substitutes-on-def3*:
bilateral-substitutes-on A f
$\longleftrightarrow (\forall\, B {\subseteq} A.\ \forall\, a {\in} A.\ \forall\, b {\in} A.\ b \notin f\ (B \cup \{b\}) \wedge b \in f\ (B \cup \{a,\ b\}) \longrightarrow Xd\ a \in Xd\ \text{‘}\ B \vee Xd\ b \in Xd\ \text{‘}\ B)$

end

As before, we define a series of locales that capture the relevant hypotheses about hospital choice functions.

locale *ContractsWithBilateralSubstitutes* = *Contracts* +
assumes *Ch-bilateral-substitutes*: $\forall\, h.$ *bilateral-substitutes* (*Ch h*)

sublocale *ContractsWithSubstitutes* < *ContractsWithBilateralSubstitutes*

locale *ContractsWithBilateralSubstitutesAndIRC* =
ContractsWithBilateralSubstitutes + *ContractsWithIRC*

sublocale *ContractsWithSubstitutesAndIRC* < *ContractsWithBilateralSubstitutesAndIRC*

context *ContractsWithBilateralSubstitutesAndIRC*
begin

The key difficulty in showing the stability of the result of the COP under this condition (Hatfield and Kojima 2010, Theorem 1) is in proving that it ensures we get an *allocation*; the remainder of the proof of §5.7 (for a single hospital, where this property is trivial) goes through unchanged. We avail ourselves of Hirata and Kasuya (2014, Lemma), which they say is a restatement of the proof of Hatfield and Kojima (2010, Theorem 1). See also Aygün and Sönmez (2012a, Appendix A).

lemma *bilateral-substitutes-lemma*:
  assumes *Xd x* ∉ *Xd* ‘ *Ch h X*
  assumes *d* ∉ *Xd* ‘ *Ch h X*
  assumes *d* ≠ *Xd x*
  shows *d* ∉ *Xd* ‘ *Ch h* (*insert x X*)
proof(*rule notI*)
  assume *d* ∈ *Xd* ‘ *Ch h* (*insert x X*)
  then obtain *x′* **where** *x′*: *x′* ∈ *Ch h* (*insert x X*) *Xd x′* = *d* **by** *blast*
  with *Ch-irc* ‹*d* ∉ *Xd* ‘ *Ch h X*›
  have *x* ∈ *Ch h* (*insert x X*) **unfolding** *irc-def* **by** *blast*
  let *?X′* = {*y* ∈ *X*. *Xd y* ∉ {*Xd x*, *d*}}
  from *Ch-range* ‹*Xd x* ∉ *Xd* ‘ *Ch h X*› ‹*d* ∉ *Xd* ‘ *Ch h X*› ‹*d* ≠ *Xd x*› *x′*
  have *Ch h* (*insert x′ ?X′*) = *Ch h X*
    using *consistencyD*[*OF Ch-consistency*[**where** *h*=*h*], **where** *B*=*X* **and** *C*=*insert x′ ?X′*]
    by (*fastforce iff*: *image-iff*)
  moreover from *Ch-range Ch-singular* ‹*d* ∉ *Xd* ‘ *Ch h X*› *x′* ‹*x* ∈ *Ch h* (*insert x X*)›
  have *Ch h* (*insert x* (*insert x′ ?X′*)) = *Ch h* (*insert x X*)
    using *consistencyD*[*OF Ch-consistency*[**where** *h*=*h*], **where** *B*=*insert x X* **and** *C*=*insert x* (*insert x′ ?X′*)]
    by (*clarsimp simp*: *insert-commute*) (*blast dest*: *inj-onD*)
  moreover note ‹*d* ∉ *Xd* ‘ *Ch h X*› *x′*
  ultimately show *False*
    using *bilateral-substitutesD*[*OF spec*[*OF Ch-bilateral-substitutes*, *of h*], **where** *a*=*x* **and** *b*=*x′* **and** *B*=*?X′*] **by** *fastforce*
qed

Our proof essentially adds the inductive details these earlier efforts skipped over. It is somewhat complicated by our use of the simultaneous-offer COP.

lemma *bilateral-substitutes-lemma-union*:
  assumes *Xd* ‘ *Ch h X* ∩ *Xd* ‘ *Y* = {}
  assumes *d* ∉ *Xd* ‘ *Ch h X*
  assumes *d* ∉ *Xd* ‘ *Y*
  assumes *allocation Y*
  shows *d* ∉ *Xd* ‘ *Ch h* (*X* ∪ *Y*)

lemma *cop-F-CH-CD-on-disjoint*:
  assumes *cop-F-closed-inv ds fp*
  assumes *cop-F-range-inv ds fp*
  shows *Xd* ‘ *CH fp* ∩ *Xd* ‘ (*CD-on ds* (− *RH fp*) − *fp*) = {}

Our key lemma shows that we effectively have *substitutes* for rejected contracts, provided the relevant doctor does not have a contract held with the relevant hospital. Note the similarity to Theorem 4 (§6.3).

lemma *cop-F-RH-mono*:
  assumes *cop-F-closed-inv ds fp*
  assumes *cop-F-range-inv ds fp*
  assumes *Xd x* ∉ *Xd* ‘ *Ch* (*Xh x*) *fp*
  assumes *x* ∈ *RH fp*
  shows *x* ∈ *RH* (*cop-F ds fp*)
proof(*safe*)
  from ‹*x* ∈ *RH fp*› **show** *x* ∈ *cop-F ds fp* **using** *cop-F-increasing* **by** *blast*
next
  assume *x* ∈ *CH* (*cop-F ds fp*)
  from *Ch-singular* ‹*x* ∈ *CH* (*cop-F ds fp*)› ‹*x* ∈ *RH fp*›
  have *Ch* (*Xh x*) (*cop-F ds fp*) = *Ch* (*Xh x*) (*fp* ∪ (*CD-on ds* (−*RH fp*) − *fp* − {*z*. *Xd z* = *Xd x*}))
    unfolding *cop-F-def*
    by − (*rule consistencyD*[*OF Ch-consistency*], *auto simp*: *mem-CH-Ch dest*: *Ch-range′ inj-onD*)
  with *cop-F-CH-CD-on-disjoint*[*OF* ‹*cop-F-closed-inv ds fp*› ‹*cop-F-range-inv ds fp*›]
  have *Xd x* ∉ *Xd* ‘ *Ch* (*Xh x*) (*cop-F ds fp*)

```
    by simp (rule bilateral-substitutes-lemma-union[OF - ‹Xd x ∉ Xd ‘ Ch (Xh x) fp›],
             auto simp: CH-def CD-on-inj-on-Xd inj-on-diff)
  with ‹x ∈ CH (cop-F ds fp)› show False by (simp add: mem-CH-Ch)
qed

lemma cop-F-allocation-inv:
  valid ds (λds fp. cop-F-range-inv ds fp ∧ cop-F-closed-inv ds fp) (λds fp. allocation (CH fp))
proof(induct rule: validI)
  case base show ?case by (simp add: CH-simps)
next
  case (step fp)
  then have allocation (CH fp)
        and cop-F-closed-inv ds fp
        and cop-F-range-inv ds fp by blast+
  note cop-F-CH-CD-on-disjoint = cop-F-CH-CD-on-disjoint[OF ‹cop-F-closed-inv ds fp› ‹cop-F-range-inv ds fp›]
  note cop-F-RH-mono = cop-F-RH-mono[OF ‹cop-F-closed-inv ds fp› ‹cop-F-range-inv ds fp›]
  show ?case
  proof(rule inj-onI)
    fix x y
    assume x ∈ CH (cop-F ds fp) and y ∈ CH (cop-F ds fp) and Xd x = Xd y
    show x = y
    proof(cases Xh y = Xh x)
      case True with Ch-singular ‹x ∈ CH (cop-F ds fp)› ‹y ∈ CH (cop-F ds fp)› ‹Xd x = Xd y›
      show ?thesis by (fastforce simp: mem-CH-Ch dest: inj-onD)
    next
      case False note ‹Xh y ≠ Xh x›
      from ‹x ∈ CH (cop-F ds fp)› show ?thesis
      proof(cases x rule: CH-cop-F-cases)
        case CH note ‹x ∈ CH fp›
        from ‹y ∈ CH (cop-F ds fp)› show ?thesis
        proof(cases y rule: CH-cop-F-cases)
          case CH note ‹y ∈ CH fp›
          with ‹allocation (CH fp)› ‹Xd x = Xd y› ‹x ∈ CH fp›
          show ?thesis by (blast dest: inj-onD)
        next
          case RH-fp note ‹y ∈ RH fp›
          from ‹allocation (CH fp)› ‹Xd x = Xd y› ‹Xh y ≠ Xh x› ‹x ∈ CH fp› have Xd y ∉ Xd ‘ Ch (Xh y) fp
            by clarsimp (metis Ch-CH-irc-idem Ch-range' inj-on-contraD)
          with ‹y ∈ CH (cop-F ds fp)› ‹y ∈ RH fp› cop-F-RH-mono show ?thesis by blast
        next
          case CD-on note y' = ‹y ∈ CD-on ds (− RH fp) − fp›
          with cop-F-CH-CD-on-disjoint ‹Xd x = Xd y› ‹x ∈ CH fp› show ?thesis by blast
        qed
      next
        case RH-fp note ‹x ∈ RH fp›
        from ‹y ∈ CH (cop-F ds fp)› show ?thesis
        proof(cases y rule: CH-cop-F-cases)
          case CH note ‹y ∈ CH fp›
          from ‹allocation (CH fp)› ‹Xd x = Xd y› ‹Xh y ≠ Xh x› ‹y ∈ CH fp› have Xd x ∉ Xd ‘ Ch (Xh x) fp
            by clarsimp (metis Ch-CH-irc-idem Ch-range' inj-on-contraD)
          with ‹x ∈ CH (cop-F ds fp)› ‹x ∈ RH fp› cop-F-RH-mono show ?thesis by blast
        next
          case RH-fp note ‹y ∈ RH fp›
          show ?thesis
          proof(cases Xd x ∈ Xd ‘ Ch (Xh x) fp)
            case True
            with ‹allocation (CH fp)› ‹Xd x = Xd y› ‹Xh y ≠ Xh x› have Xd y ∉ Xd ‘ Ch (Xh y) fp
              by clarsimp (metis Ch-range' inj-onD mem-CH-Ch)
```

```
          with ‹y ∈ CH (cop-F ds fp)› ‹y ∈ RH fp› cop-F-RH-mono show ?thesis by blast
        next
          case False note ‹Xd x ∉ Xd ‘ Ch (Xh x) fp›
          with ‹x ∈ CH (cop-F ds fp)› ‹x ∈ RH fp› cop-F-RH-mono show ?thesis by blast
        qed
      next
        case CD-on note ‹y ∈ CD-on ds (− RH fp) − fp›
        from cop-F-CH-CD-on-disjoint ‹Xd x = Xd y› ‹y ∈ CD-on ds (− RH fp) − fp›
        have Xd x ∉ Xd ‘ Ch (Xh x) fp by (auto simp: CH-def dest: Ch-range′)
        with ‹x ∈ CH (cop-F ds fp)› ‹x ∈ RH fp› cop-F-RH-mono show ?thesis by blast
      qed
    next
      case CD-on note ‹x ∈ CD-on ds (− RH fp) − fp›
      from ‹y ∈ CH (cop-F ds fp)› show ?thesis
      proof(cases y rule: CH-cop-F-cases)
        case CH note ‹y ∈ CH fp›
        with cop-F-CH-CD-on-disjoint ‹Xd x = Xd y› ‹x ∈ CD-on ds (− RH fp) − fp› show ?thesis by blast
      next
        case RH-fp note ‹y ∈ RH fp›
        from cop-F-CH-CD-on-disjoint ‹Xd x = Xd y› ‹x ∈ CD-on ds (− RH fp) − fp›
        have Xd y ∉ Xd ‘ Ch (Xh y) fp unfolding CH-def by clarsimp (blast dest: Ch-range′)
        with ‹y ∈ CH (cop-F ds fp)› ‹y ∈ RH fp› cop-F-RH-mono show ?thesis by blast
      next
        case CD-on note ‹y ∈ CD-on ds (− RH fp) − fp›
        with ‹Xd x = Xd y› ‹x ∈ CD-on ds (− RH fp) − fp› show ?thesis
          by (meson CD-on-inj-on-Xd DiffD1 inj-on-eq-iff)
      qed
    qed
  qed
 qed
qed

lemma fp-cop-F-allocation:
 shows allocation (cop ds)

theorem Theorem-1:
 shows stable-on ds (cop ds)

end
```

Hatfield and Kojima (2010, §3.1) provide an example that shows that the traditional optimality and strategic results do not hold under *bilateral-substitutes*, which motivates looking for a stronger condition that remains weaker than *substitutes*.

Their example involves two doctors, two hospitals, and five contracts.

```
datatype X5 = Xd1 | Xd1′ | Xd2 | Xd2′ | Xd2″

primrec X5d :: X5 ⇒ D2 where
  X5d Xd1 = D1
| X5d Xd1′ = D1
| X5d Xd2 = D2
| X5d Xd2′ = D2
| X5d Xd2″ = D2

primrec X5h :: X5 ⇒ H2 where
  X5h Xd1 = H1
| X5h Xd1′ = H1
| X5h Xd2 = H1
| X5h Xd2′ = H2
```

| $X5h\ Xd2'' = H1$

primrec $PX5d :: D2 \Rightarrow X5\ rel$ **where**
  $PX5d\ D1 = linord\text{-}of\text{-}list\ [Xd1,\ Xd1']$
| $PX5d\ D2 = linord\text{-}of\text{-}list\ [Xd2,\ Xd2',\ Xd2'']$

primrec $CX5h :: H2 \Rightarrow X5\ cfun$ **where**
  $CX5h\ H1\ A =$
    $(if\ \{Xd1',\ Xd2\} \subseteq A\ then\ \{Xd1',\ Xd2\}\ else$
     $if\ \{Xd2''\} \subseteq A\ then\ \{Xd2''\}\ else$
     $if\ \{Xd1\} \subseteq A\ then\ \{Xd1\}\ else$
     $if\ \{Xd1'\} \subseteq A\ then\ \{Xd1'\}\ else$
     $if\ \{Xd2\} \subseteq A\ then\ \{Xd2\}\ else\ \{\})$
| $CX5h\ H2\ A = \{\ x\ .\ x \in A \land x = Xd2'\ \}$

interpretation BSI: $Contracts\ X5d\ X5h\ PX5d\ CX5h$

lemma $CX5h\text{-}bilateral\text{-}substitutes$:
  shows $BSI.bilateral\text{-}substitutes\ (CX5h\ h)$
unfolding $BSI.bilateral\text{-}substitutes\text{-}def$ **by** $(cases\ h)\ (auto\ simp\text{:}\ X5\text{-}ALL)$

lemma $CX5h\text{-}irc$:
  shows $irc\ (CX5h\ h)$
unfolding $irc\text{-}def$ **by** $(cases\ h)\ (auto\ simp\text{:}\ X5\text{-}ALL)$

interpretation BSI: $ContractsWithBilateralSubstitutesAndIRC\ X5d\ X5h\ PX5d\ CX5h$

There are two stable matches in this model.

lemma $BSI\text{-}stable$:
  shows $BSI.stable\ X \longleftrightarrow X = \{Xd1,\ Xd2''\} \lor X = \{Xd1',\ Xd2\}$

Therefore there is no doctor-optimal match under these preferences:

lemma
  $\neg(\exists\ (Y{::}X5\ set).\ BSI.doctor\text{-}optimal\text{-}match\ UNIV\ Y)$
unfolding $BSI.doctor\text{-}optimal\text{-}match\text{-}def\ BSI\text{-}stable$
apply $clarsimp$
apply $(cut\text{-}tac\ X{=}Y$ **in** $X5\text{-}pow)$
apply $clarsimp$
apply $(elim\ disjE;\ simp\ add\text{:}\ insert\text{-}eq\text{-}iff;\ simp\ add\text{:}\ X5\text{-}ALL\ linord\text{-}of\text{-}list\text{-}linord\text{-}of\text{-}listP)$
done

6.2 Theorem 3: *pareto separability* relates *unilateral substitutes* and *substitutes*

Hatfield and Kojima (2010, §4) proceed to define *unilateral substitutes*:

> [P]references satisfy *unilateral substitutes* if whenever a hospital rejects the contract z when that is the only contract with $Xd\ z$ available, it still rejects the contract z when the choice set expands.

context $Contracts$
begin

definition $unilateral\text{-}substitutes\text{-}on :: {}'x\ set \Rightarrow {}'x\ cfun \Rightarrow bool$ **where**
  $unilateral\text{-}substitutes\text{-}on\ A\ f$
    $\longleftrightarrow \neg(\exists\, B{\subseteq}A.\ \exists\, a\ b.\ \{a,\ b\} \subseteq A \land Xd\ b \notin Xd\ `\ B \land b \notin f\ (B \cup \{b\}) \land b \in f\ (B \cup \{a,\ b\}))$

abbreviation $unilateral\text{-}substitutes :: {}'x\ cfun \Rightarrow bool$ **where**

unilateral-substitutes ≡ *unilateral-substitutes-on UNIV*

lemma *unilateral-substitutes-on-def2*:
unilateral-substitutes-on A f
$\longleftrightarrow (\forall B{\subseteq}A.\ \forall a{\in}A.\ \forall b{\in}A.\ Xd\ b \notin Xd\ `\ B \wedge b \notin f\ (B \cup \{b\}) \longrightarrow b \notin f\ (B \cup \{a,\ b\}))$

Aygün and Sönmez (2012a, §4, Definition 6) give the following equivalent definition:

lemma *unilateral-substitutes-on-def3*:
unilateral-substitutes-on A f
$\longleftrightarrow (\forall B{\subseteq}A.\ \forall a{\in}A.\ \forall b{\in}A.\ b \notin f\ (B \cup \{b\}) \wedge b \in f\ (B \cup \{a,\ b\}) \longrightarrow Xd\ b \in Xd\ `\ B)$

lemma *substitutes-on-unilateral-substitutes-on*:
assumes *substitutes-on A f*
shows *unilateral-substitutes-on A f*

lemma *unilateral-substitutes-on-bilateral-substitutes-on*:
assumes *unilateral-substitutes-on A f*
shows *bilateral-substitutes-on A f*

The following defines locales for the *unilateral-substitutes* hypothesis, and inserts these between those for *substitutes* and *bilateral-substitutes*.

end

locale *ContractsWithUnilateralSubstitutes* = *Contracts* +
assumes *Ch-unilateral-substitutes*: ∀ *h*. *unilateral-substitutes* (*Ch h*)

sublocale *ContractsWithUnilateralSubstitutes* < *ContractsWithBilateralSubstitutes*

sublocale *ContractsWithSubstitutes* < *ContractsWithUnilateralSubstitutes*

locale *ContractsWithUnilateralSubstitutesAndIRC* =
ContractsWithUnilateralSubstitutes + *ContractsWithIRC*

sublocale *ContractsWithUnilateralSubstitutesAndIRC* < *ContractsWithBilateralSubstitutesAndIRC*

sublocale *ContractsWithSubstitutesAndIRC* < *ContractsWithUnilateralSubstitutesAndIRC*

Hatfield and Kojima (2010, Theorem 3) relate *unilateral-substitutes* to *substitutes* using *Pareto separability*:

> Preferences are *Pareto separable* for a hospital if the hospital's choice between x and x', two [distinct] contracts with the same doctor, does not depend on what other contracts the hospital has access to.

This result also depends on *irc*.

context *Contracts*
begin

definition *pareto-separable-on* :: *'x set* ⇒ *bool* **where**
pareto-separable-on A
$\longleftrightarrow (\forall B{\subseteq}A.\ \forall C{\subseteq}A.\ \forall a\ b.\ \{a,\ b\} \subseteq A \wedge a \neq b \wedge Xd\ a = Xd\ b \wedge Xh\ a = Xh\ b$
$\wedge\ a \in Ch\ (Xh\ b)\ (B \cup \{a,\ b\}) \longrightarrow b \notin Ch\ (Xh\ b)\ (C \cup \{a,\ b\}))$

abbreviation *pareto-separable* :: *bool* **where**
pareto-separable ≡ *pareto-separable-on UNIV*

lemma *substitutes-on-pareto-separable-on*:
assumes ∀ *h*. *substitutes-on A* (*Ch h*)

```
  shows pareto-separable-on A
proof(rule pareto-separable-onI)
  fix B C a b
  assume XXX: B ⊆ A C ⊆ A a ∈ A b ∈ A a ≠ b Xd a = Xd b Xh a = Xh b a ∈ Ch (Xh b) (insert a (insert b
B))
   note Ch-iiaD = iia-onD[OF iffD1[OF substitutes-iia spec[OF ‹∀ h. substitutes-on A (Ch h)›]], rotated −1,
simplified]
  from XXX have a ∈ Ch (Xh b) {a, b} by (fast elim: Ch-iiaD)
  with XXX have b ∉ Ch (Xh b) {a, b} by (meson Ch-singular inj-on-eq-iff)
  with XXX have b ∉ Ch (Xh b) (C ∪ {a, b}) by (blast dest: Ch-iiaD)
  with XXX show b ∉ Ch (Xh b) (insert a (insert b C)) by simp
qed

lemma unilateral-substitutes-on-pareto-separable-on-substitutes-on:
  assumes ∀ h. unilateral-substitutes-on A (Ch h)
  assumes ∀ h. irc-on A (Ch h)
  assumes pareto-separable-on A
  shows substitutes-on A (Ch h)
proof(rule substitutes-onI)
  fix B a b
  assume XXX: B ⊆ A a ∈ A b ∈ A b ∉ Ch h (insert b B)
  show b ∉ Ch h (insert a (insert b B))
  proof(cases Xd b ∈ Xd ‘ B)
    case True show ?thesis
    proof(cases Xd b ∈ Xd ‘ Ch h (insert b B))
      case True
      then obtain x where x ∈ Ch h (insert b B) Xd x = Xd b by force
      moreover with XXX have x ∈ B x ≠ b using Ch-range by blast+
      moreover note ‹pareto-separable-on A› XXX
      ultimately show ?thesis
        using pareto-separable-onD[where A=A and B=B − {x} and a=x and b=b and C=insert a (B − {x})]
Ch-range
        by (cases Xh b = h) (auto simp: insert-commute insert-absorb)
    next
      case False
      let ?B′ = {x ∈ B . Xd x ≠ Xd b}
      from False have b ∉ Ch h (insert b B) by blast
      with ‹∀ h. irc-on A (Ch h)› XXX False have b ∉ Ch h (insert b ?B′)
        using consistency-onD[OF irc-on-consistency-on[where f=Ch h], where B=insert b B and C=insert b
?B′] Ch-range
        by (fastforce iff: image-iff)
      with ‹∀ h. unilateral-substitutes-on A (Ch h)› XXX False have b ∉ Ch h (insert a (insert b ?B′))
        using unilateral-substitutes-onD[where f=Ch h and B=?B′]
        by (fastforce iff: image-iff)
      with ‹∀ h. irc-on A (Ch h)› XXX False show ?thesis
        using consistency-onD[OF irc-on-consistency-on[where f=Ch h],
                          where A=A and B=insert a (insert b B) and C=insert a (insert b ?B′)]
              Ch-range′[of - h insert a (insert b B)] Ch-singular
        by simp (blast dest: inj-on-contraD)
    qed
  next
    case False
    with ‹∀ h. unilateral-substitutes-on A (Ch h)› XXX show ?thesis by (blast dest: unilateral-substitutes-onD)
  qed
qed

theorem Theorem-3:
  assumes ∀ h. irc-on A (Ch h)
```

shows $(\forall h.\ \textit{substitutes-on}\ A\ (\textit{Ch}\ h)) \longleftrightarrow (\forall h.\ \textit{unilateral-substitutes-on}\ A\ (\textit{Ch}\ h) \wedge \textit{pareto-separable-on}\ A)$

end

6.2.1 Afacan and Turhan (2015): *doctor separability* relates bi- and unilateral substitutes

context *Contracts*
begin

Afacan and Turhan (2015, Theorem 1) relate *bilateral-substitutes* and *unilateral-substitutes* using *doctor separability*:

> *[Doctor separability (DS)]* says that if a doctor is not chosen from a set of contracts in the sense that no contract of him is selected, then that doctor should still not be chosen unless a contract of a new doctor (that is, doctor having no contract in the given set of contracts) becomes available. For practical purposes, we can consider DS as capturing contracts where certain groups of doctors are substitutes. [footnote: If $Xd\ x \notin Xd$ ‘ $Ch\ h\ (Y \cup \{x,\ z\})$, then doctor $Xd\ x$ is not chosen. And under DS, he continues not to be chosen unless a new doctor comes. Hence, we can interpret it as the doctors in the given set of contracts are substitutes.]

definition *doctor-separable-on* :: $'x\ set \Rightarrow 'x\ cfun \Rightarrow bool$ **where**
  doctor-separable-on $A\ f$
    $\longleftrightarrow (\forall B{\subseteq}A.\ \forall a\ b\ c.\ \{a,\ b,\ c\} \subseteq A \wedge Xd\ a \neq Xd\ b \wedge Xd\ b = Xd\ c \wedge Xd\ a \notin Xd$ ‘ $f\ (B \cup \{a,\ b\})$
      $\longrightarrow Xd\ a \notin Xd$ ‘ $f\ (B \cup \{a,\ b,\ c\}))$

abbreviation *doctor-separable* :: $'x\ cfun \Rightarrow bool$ **where**
  doctor-separable $\equiv$ *doctor-separable-on UNIV*

lemma *unilateral-substitutes-on-doctor-separable-on*:
  assumes *unilateral-substitutes-on* $A\ f$
  assumes *irc-on* $A\ f$
  assumes $\forall B{\subseteq}A.$ *allocation* $(f\ B)$
  assumes *f-range-on* $A\ f$
  shows *doctor-separable-on* $A\ f$
proof(*rule doctor-separable-onI*)
  fix $B\ a\ b\ c$
  assume *XXX*: $B \subseteq A\ a \in A\ b \in A\ c \in A\ Xd\ a \neq Xd\ b\ Xd\ b = Xd\ c\ Xd\ a \notin Xd$ ‘ f (*insert* a (*insert* $b\ B$))
  have $a \notin f$ (*insert* a (*insert* b (*insert* $c\ B$)))
  proof(*rule notI*)
    assume *a*: $a \in f$ (*insert* a (*insert* b (*insert* $c\ B$)))
    let $?C = \{x \in B\ .\ Xd\ x \neq Xd\ a \vee x = a\}$
    from ‹*irc-on* $A\ f$› ‹*f-range-on* $A\ f$› *XXX*(*1−3*,*7*)
    have f (*insert* a (*insert* $b\ B$)) $= f$ (*insert* a (*insert* $b\ ?C$))
      by − (*rule consistency-onD*[*OF irc-on-consistency-on*[**where** $A{=}A$ **and** $f{=}f$]];
          fastforce dest: *f-range-onD*[**where** $A{=}A$ **and** $f{=}f$ **and** $B{=}$*insert* a (*insert* $b\ B$)] *simp*: *rev-image-eqI*)
    with ‹*unilateral-substitutes-on* $A\ f$› *XXX*
    have *abcC*: $a \notin f$ (*insert* a (*insert* b (*insert* $c\ ?C$)))
      using *unilateral-substitutes-onD*[**where** $A{=}A$ **and** $f{=}f$ **and** $a{=}c$ **and** $b{=}a$ **and** $B{=}$*insert* $b\ ?C - \{a\}$]
      by (*force simp*: *insert-commute*)
    from ‹*irc-on* $A\ f$› ‹$\forall B{\subseteq}A.$ *allocation* $(f\ B)$› ‹*f-range-on* $A\ f$› *XXX*(*1−4*) *a*
    have f (*insert* a (*insert* b (*insert* $c\ B$))) $= f$ (*insert* a (*insert* b (*insert* $c\ ?C$)))
      by − (*rule consistency-onD*[*OF irc-on-consistency-on*[**where** $A{=}A$ **and** $f{=}f$]], (*auto*)[*4*],
          clarsimp, *rule conjI*, *blast dest*!: *f-range-onD'*[**where** $A{=}A$], *metis inj-on-contraD insert-subset*)
    with *a abcC* **show** *False* **by** *simp*
  qed
  moreover
  have $a' \notin f$ (*insert* a (*insert* b (*insert* $c\ B$))) **if** a': $a' \in B\ Xd\ a' = Xd\ a$ **for** a'
  proof(*rule notI*)
    assume $a'X$: $a' \in f$ (*insert* a (*insert* b (*insert* $c\ B$)))
    let $?B =$ *insert* $a\ B - \{a'\}$

```
    from XXX a′
    have XXX-7′: Xd a ∉ Xd ‘ f (insert a′ (insert b ?B))
      by clarsimp (metis imageI insert-Diff-single insert-absorb insert-commute)
    let ?C = {x ∈ ?B . Xd x ≠ Xd a ∨ x = a′}
    from ‹irc-on A f› ‹f-range-on A f› XXX(1−3) a′ XXX-7′
    have f (insert a′ (insert b ?B)) = f (insert a′ (insert b ?C))
      by − (rule consistency-onD[OF irc-on-consistency-on[where A=A and f=f]];
            fastforce dest: f-range-onD[where A=A and f=f and B=insert a′ (insert b ?B)] simp: rev-image-eqI)
    with ‹unilateral-substitutes-on A f› XXX(1−6) XXX-7′ a′
    have abcC: a′ ∉ f (insert a′ (insert b (insert c ?C)))
      using unilateral-substitutes-onD[where A=A and f=f and a=c and b=a′ and B=insert b ?C − {a′}]
      by (force simp: insert-commute rev-image-eqI)
    have f (insert a′ (insert b (insert c ?B))) = f (insert a′ (insert b (insert c ?C)))
    proof(rule consistency-onD[OF irc-on-consistency-on[where A=A and f=f]])
      from a′ have insert a′ (insert b (insert c ?B)) = insert a (insert b (insert c B)) by blast
      with ‹∀ B⊆A. allocation (f B)› ‹f-range-on A f› XXX(1−4) a′ a′X
      show f (insert a′ (insert b (insert c ?B))) ⊆ insert a′ (insert b (insert c {x ∈ ?B. Xd x ≠ Xd a ∨ x = a′}))
        by clarsimp (rule conjI, blast dest!: f-range-onD′[where A=A], metis inj-on-contraD insert-subset)
    qed (use ‹irc-on A f› XXX(1−4) a′ in auto)
    with a′ a′X abcC show False by simp (metis insert-Diff insert-Diff-single insert-commute)
  qed
  moreover note ‹f-range-on A f› XXX
  ultimately show Xd a ∉ Xd ‘ f (insert a (insert b (insert c B)))
    by (fastforce dest: f-range-onD[where B=insert a (insert b (insert c B))])
qed

lemma bilateral-substitutes-on-doctor-separable-on-unilateral-substitutes-on:
  assumes bilateral-substitutes-on A f
  assumes doctor-separable-on A f
  assumes f-range-on A f
  shows unilateral-substitutes-on A f
proof(rule unilateral-substitutes-onI)
  fix B a b
  assume XXX: B ⊆ A a ∈ A b ∈ A Xd b ∉ Xd ‘ B b ∉ f (insert b B)
  show b ∉ f (insert a (insert b B))
  proof(cases Xd a ∈ Xd ‘ B)
    case True
    then obtain C c where Cc: B = insert c C c ∉ C Xd c = Xd a by (metis Set.set-insert image-iff)
    from ‹b ∉ f (insert b B)› Cc have b ∉ f (insert b (insert c C)) by simp
    with ‹f-range-on A f› XXX Cc have Xd b ∉ Xd ‘ f (insert b (insert c C))
      by clarsimp (metis f-range-onD′ image-eqI insertE insert-subset)
    with ‹doctor-separable-on A f› XXX Cc show ?thesis
      by (auto simp: insert-commute dest: doctor-separable-onD)
  qed (use ‹bilateral-substitutes-on A f› XXX in ‹simp add: bilateral-substitutes-onD›)
qed

theorem unilateral-substitutes-on-doctor-separable-on-bilateral-substitutes-on:
  assumes irc-on A f
  assumes ∀ B⊆A. allocation (f B) — A rephrasing of Ch-singular.
  assumes f-range-on A f
  shows unilateral-substitutes-on A f ⟷ bilateral-substitutes-on A f ∧ doctor-separable-on A f
```

Afacan and Turhan (2015, Remark 2) observe the independence of the *doctor-separable*, *pareto-separable* and *bilateral-substitutes* conditions.

end

6.3 Theorems 4 and 5: Doctor optimality

context *ContractsWithUnilateralSubstitutesAndIRC*
begin

We return to analyzing the COP following Hatfield and Kojima (2010). The next goal is to establish a doctor-optimality result for it in the spirit of §5.3.

We first show that, with hospital choice functions satisfying *unilateral-substitutes*, we effectively have the *substitutes* condition for all contracts that have been rejected. In other words, hospitals never renegotiate with doctors.

The proof is by induction over the finite set Y.

lemma
  assumes $Xd\ x \notin Xd\ `\ Ch\ h\ X$
  assumes $x \in X$
  shows *no-renegotiation-union*: $x \notin Ch\ h\ (X \cup Y)$
    and $x \notin Ch\ h\ (insert\ x\ ((X \cup Y) - \{z.\ Xd\ z = Xd\ x\}))$

To discharge the first antecedent of this lemma, we need an invariant for the COP that asserts that, for each doctor d, there is a subset of the contracts currently offered by d that was previously uniformly rejected by the COP, for each contract that is rejected at the current step. To support a later theorem (see §6.3) we require these subsets to be upwards-closed with respect to the doctor's preferences.

definition
  cop-F-rejected-inv :: $'b\ set \Rightarrow\ 'a\ set \Rightarrow bool$
where
  cop-F-rejected-inv ds fp $\longleftrightarrow (\forall x{\in}RH\ fp.\ \exists fp'{\subseteq}fp.\ x \in fp' \wedge above\ (Pd\ (Xd\ x))\ x \subseteq fp' \wedge Xd\ x \notin Xd\ `\ CH\ fp')$

lemma *cop-F-rejected-inv*:
  shows *valid ds* (λ*ds fp. cop-F-range-inv ds fp* $\wedge$ *cop-F-closed-inv ds fp* $\wedge$ *allocation* (*CH fp*)) *cop-F-rejected-inv*

lemma *fp-cop-F-rejected-inv*:
  shows *cop-F-rejected-inv ds* (*fp-cop-F ds*)

Hatfield and Kojima (2010, Theorem 4) assert that we effectively recover *substitutes* for the contracts relevant to the COP. We cannot adopt their phrasing as it talks about the execution traces of the COP, and not just its final state. Instead we present the result we use, which relates two consecutive states in an execution trace of the COP:

theorem *Theorem-4*:
  assumes *cop-F-rejected-inv ds fp*
  assumes $x \in RH\ fp$
  shows $x \in RH$ (*cop-F ds fp*)

Another way to interpret *cop-F-rejected-inv* is to observe that the doctor-optimal match contains the least preferred of the contracts that the doctors have offered.

corollary *fp-cop-F-worst*:
  assumes $x \in cop\ ds$
  assumes $y \in$ *fp-cop-F ds*
  assumes $Xd\ y = Xd\ x$
  shows $(x,\ y) \in Pd\ (Xd\ x)$

The doctor optimality result, Theorem 5, hinges on showing that no contract in any stable match is ever rejected.

definition
  theorem-5-inv :: $'b\ set \Rightarrow\ 'a\ set \Rightarrow bool$
where
  theorem-5-inv ds fp $\longleftrightarrow RH\ fp \cap \bigcup\{X.\ stable\text{-}on\ ds\ X\} = \{\}$

lemma *theorem-5-inv*:

```
  shows valid ds (λds fp. cop-F-range-inv ds fp ∧ cop-F-closed-inv ds fp
                   ∧ allocation (CH fp) ∧ cop-F-rejected-inv ds fp) theorem-5-inv
proof(induct rule: validI)
  case base show ?case unfolding theorem-5-inv-def by simp
next
  case (step fp)
  then have cop-F-range-inv ds fp
       and cop-F-closed-inv ds fp
       and allocation (CH fp)
       and cop-F-rejected-inv ds fp
       and theorem-5-inv ds fp by blast+
  show ?case
  proof(rule theorem-5-invI)
    fix z X assume z: z ∈ RH (cop-F ds fp) and z ∈ X and stable-on ds X
    from ‹theorem-5-inv ds fp› ‹z ∈ X› ‹stable-on ds X›
    have z': z ∉ RH fp unfolding theorem-5-inv-def by blast
    define Y where Y ≡ Ch (Xh z) (cop-F ds fp)
    from z have YYY: z ∉ Ch (Xh z) (insert z Y)
      using consistencyD[OF Ch-consistency]
      by (simp add: mem-CH-Ch Y-def)
        (metis Ch-f-range f-range-on-def insert-subset subset-insertI top-greatest)
    have yRx: (x, y) ∈ Pd (Xd y) if x ∈ X and y ∈ Y and Xd y = Xd x for x y
    proof(rule ccontr)
      assume (x, y) ∉ Pd (Xd y)
      with Pd-linear ‹cop-F-range-inv ds fp› ‹stable-on ds X› that
      have BBB: (y, x) ∈ Pd (Xd y) ∧ x ≠ y
        unfolding Y-def cop-F-def cop-F-range-inv-def order-on-defs total-on-def
        by (clarsimp simp: mem-CD-on-Cd dest!: Ch-range') (metis Cd-range' Int-iff refl-onD stable-on-range')
      from ‹stable-on ds X› ‹cop-F-closed-inv ds fp› ‹theorem-5-inv ds fp› BBB that have x ∈ fp ∧ y ∈ fp
        unfolding cop-F-def cop-F-closed-inv-def theorem-5-inv-def above-def Y-def
        by (fastforce simp: mem-CD-on-Cd dest: Ch-range' Cd-preferred)
      with ‹stable-on ds X› ‹theorem-5-inv ds fp› ‹x ∈ X› have x ∈ Ch (Xh x) fp
        unfolding theorem-5-inv-def by (force simp: mem-CH-Ch)
      with ‹allocation (CH fp)› ‹Xd y = Xd x› BBB have y ∉ Ch (Xh z) fp
        by (metis Ch-range' inj-onD mem-CH-Ch)
      with ‹y ∈ Y› ‹x ∈ fp ∧ y ∈ fp› show False
        unfolding Y-def using Theorem-4[OF ‹cop-F-rejected-inv ds fp›, where x=y]
        by (metis Ch-range' Diff-iff mem-CH-Ch)
    qed
    have Xd z ∉ Xd ‘ Y
    proof(safe)
      fix w assume w: Xd z = Xd w w ∈ Y
      show False
      proof(cases z ∈ fp)
        case True note ‹z ∈ fp›
        show False
        proof(cases w ∈ fp)
          case True note ‹w ∈ fp›
          from ‹Xd z = Xd w› ‹w ∈ Y› z' ‹z ∈ fp› have w ∉ CH fp
            by (metis Ch-irc-idem DiffI YYY Y-def ‹allocation (CH fp)› inj-on-eq-iff insert-absorb)
          with Theorem-4[OF ‹cop-F-rejected-inv ds fp›, where x=w] ‹w ∈ Y› ‹w ∈ fp› show False
            unfolding Y-def CH-def by simp
        next
          case False note ‹w ∉ fp›
          with ‹w ∈ Y› have w ∉ Ch (Xh z) fp ∧ w ∈ cop-F ds fp ∧ Xh w = Xh z
            unfolding Y-def by (blast dest: Ch-range')
          with ‹cop-F-closed-inv ds fp› ‹cop-F-range-inv ds fp› ‹z ∉ RH fp› ‹w ∉ fp› ‹z ∈ fp› ‹Xd z = Xd w›
          show False
```

```
          unfolding cop-F-closed-inv-def cop-F-range-inv-def above-def
          by (fastforce simp: cop-F-def mem-CD-on-Cd Cd-greatest greatest-def)
      qed
    next
      case False note ‹z ∉ fp›
      from ‹cop-F-range-inv ds fp› ‹cop-F-closed-inv ds fp› z ‹z ∉ fp› have Xd z ∉ Xd ‘ Ch (Xh z) fp
        unfolding cop-F-range-inv-def cop-F-closed-inv-def above-def
        by (clarsimp simp: mem-CD-on-Cd Cd-greatest greatest-def dest!: mem-Ch-CH elim!: cop-F-cases)
          (blast dest: CH-range')
      with w ‹z ∈ RH (cop-F ds fp)› ‹z ∉ fp› show False
        by (clarsimp simp: Y-def cop-F-def mem-CH-Ch)
          (metis CD-on-inj-on-Xd Ch-range' Un-iff inj-onD no-renegotiation-union)
    qed
  qed
  show False
  proof(cases z ∈ Ch (Xh z) (X ∪ Y))
    case True note ‹z ∈ Ch (Xh z) (X ∪ Y)›
    with ‹z ∈ X› have Xd z ∈ Xd ‘ Ch (Xh z) (insert z Y)
      using no-renegotiation-union[where X=insert z Y and Y=X − {z} and x=z and h=Xh z]
      by clarsimp (metis Un-insert-right insert-Diff Un-commute)
    with ‹Xd z ∉ Xd ‘ Y› ‹z ∉ Ch (Xh z) (insert z Y)› show False by (blast dest: Ch-range')
  next
    case False note ‹z ∉ Ch (Xh z) (X ∪ Y)›
    have ¬stable-on ds X
    proof(rule blocking-on-imp-not-stable[OF blocking-onI])
      from False ‹z ∈ X› ‹stable-on ds X›
      show Ch (Xh z) (X ∪ Y) ≠ Ch (Xh z) X
        using mem-CH-Ch stable-on-CH by blast
      show Ch (Xh z) (X ∪ Y) = Ch (Xh z) (X ∪ Ch (Xh z) (X ∪ Y))
        using Ch-range' by (blast intro!: consistencyD[OF Ch-consistency])
    next
      fix x assume x ∈ Ch (Xh z) (X ∪ Y)
      with Ch-singular'[of Xh z X ∪ Ch (Xh z) (cop-F ds fp)]
           invariant-cop-FD[OF cop-F-range-inv ‹cop-F-range-inv ds fp›]
           stable-on-allocation[OF ‹stable-on ds X›] stable-on-Xd[OF ‹stable-on ds X›]
           stable-on-range'[OF ‹stable-on ds X›]
      show x ∈ CD-on ds (X ∪ Ch (Xh z) (X ∪ Y))
        unfolding cop-F-range-inv-def
        by (clarsimp simp: mem-CD-on-Cd Cd-greatest greatest-def)
           (metis Ch-range' IntE Pd-range' Pd-refl Un-iff Y-def inj-onD yRx)
    qed
    with ‹stable-on ds X› show False by blast
  qed
 qed
qed

lemma fp-cop-F-theorem-5-inv:
 shows theorem-5-inv ds (fp-cop-F ds)

theorem Theorem-5:
 assumes stable-on ds X
 assumes x ∈ X
 shows ∃ y ∈ cop ds. (x, y) ∈ Pd (Xd x)
proof −
 from fp-cop-F-theorem-5-inv assms
 have x: x ∉ RH (fp-cop-F ds)
   unfolding theorem-5-inv-def by blast
 show ?thesis
```

```
  proof(cases Xd x ∈ Xd ` cop ds)
    case True
    then obtain z where z: z ∈ cop ds Xd z = Xd x by auto
    show ?thesis
    proof(cases (x, z) ∈ Pd (Xd x))
      case True with z show ?thesis by blast
    next
      case False
      with Pd-linear'[where d=Xd x] fp-cop-F-range-inv'[of z ds] assms z
      have (z, x) ∈ Pd (Xd x)
        unfolding order-on-defs total-on-def by (metis CH-range' refl-onD stable-on-range')
      with fp-cop-F-closed-inv'[of z ds x] x z have x ∈ fp-cop-F ds
        unfolding above-def by (force simp: mem-CH-Ch dest: Ch-range')
      with fp-cop-F-allocation x z have z = x by (fastforce dest: inj-onD)
      with Pd-linear assms z show ?thesis
        by (meson equalityD2 stable-on-range' underS-incl-iff)
    qed
  next
    case False note ‹Xd x ∉ Xd ` cop ds›
    with assms x show ?thesis
      by (metis DiffI Diff-eq-empty-iff fp-cop-F-all emptyE imageI stable-on-Xd stable-on-range')
  qed
qed

theorem fp-cop-F-doctor-optimal-match:
  shows doctor-optimal-match ds (cop ds)

end
```

The next lemma demonstrates the *opposition of interests* of doctors and hospitals: if all doctors weakly prefer one stable match to another, then the hospitals weakly prefer the converse.

As we do not have linear preferences for hospitals, we use revealed preference and hence assume *irc* holds of hospital choice functions. Our definition of the doctor-preferred ordering *dpref* follows the Isabelle/HOL convention of putting the larger (more preferred) element on the right, and takes care with unemployment.

```
context Contracts
begin

definition dpref :: 'x set ⇒ 'x set ⇒ bool where
  dpref X Y = (∀ x∈X. ∃ y∈Y. (x, y) ∈ Pd (Xd x))

end

context ContractsWithIRC
begin

theorem Lemma-1:
  assumes stable-on ds Y
  assumes stable-on ds Z
  assumes dpref Z Y
  assumes x ∈ Ch h Z
  shows x ∈ Ch h (Y ∪ Z)
proof(rule ccontr)
  assume x ∉ Ch h (Y ∪ Z)
  from ‹x ∈ Ch h Z› ‹x ∉ Ch h (Y ∪ Z)›
  have Ch h (Y ∪ Z) ≠ Ch h Z by blast
  moreover
  have Ch h (Y ∪ Z) = Ch h (Z ∪ Ch h (Y ∪ Z))
   by (rule consistency-onD[OF Ch-consistency]; auto dest: Ch-range')
```

```
  moreover
  have y ∈ CD-on ds (Z ∪ Ch h (Y ∪ Z)) if y ∈ Ch h (Y ∪ Z) for y
  proof −
    from ‹stable-on ds Y› ‹stable-on ds Z› that
    have Xd y ∈ ds ∧ y ∈ Field (Pd (Xd y))
      using stable-on-Xd stable-on-range' Ch-range' by (meson Un-iff)
    with Pd-linear'[of Xd y] Ch-singular ‹stable-on ds Y› ‹stable-on ds Z› ‹dpref Z Y› that show ?thesis
      unfolding dpref-def
      by (clarsimp simp: mem-CD-on-Cd Cd-greatest greatest-def)
         (metis Ch-range' Pd-Xd Un-iff eq-iff inj-on-contraD stable-on-allocation underS-incl-iff)
  qed
  ultimately show False by (blast dest: stable-on-blocking-onD[OF ‹stable-on ds Z›])
qed

end
```

Hatfield and Kojima (2010, Corollary 1 (of Theorem 5 and Lemma 1)): *unilateral-substitutes* implies there is a hospital-pessimal match, which is indeed the doctor-optimal one.

```
context ContractsWithUnilateralSubstitutesAndIRC
begin

theorem Corollary-1:
  assumes stable-on ds Z
  shows dpref Z (cop ds)
    and x ∈ Z ⟹ x ∈ Ch (Xh x) (cop ds ∪ Z)
proof −
  show dpref Z (cop ds)
    by (rule dprefI[OF Theorem-5[OF ‹stable-on ds Z›]])
  fix x assume x ∈ Z with assms show x ∈ Ch (Xh x) (cop ds ∪ Z)
    using Lemma-1[OF Theorem-1 assms ‹dpref Z (cop ds)›] stable-on-CH
    by (fastforce simp: mem-CH-Ch)
qed
```

Hatfield and Kojima (2010, p1717) show that there is not always a hospital-optimal/doctor-pessimal match when hospital preferences satisfy *unilateral-substitutes*, in contrast to the situation under *substitutes* (see §5.3). This reflects the loss of the lattice structure.

```
end
```

6.4 Theorem 6: A "rural hospitals" theorem

Hatfield and Kojima (2010, Theorem 6) demonstrates a "rural hospitals" theorem for the COP assuming hospital choice functions satisfy *unilateral-substitutes* and *lad*, as for §5.6. However Aygün and Sönmez (2012a, §4, Example 1) observe that *lad-on-substitutes-on-irc-on* does not hold with *bilateral-substitutes* instead of *substitutes*, and their Example 3 similarly for *unilateral-substitutes*. Moreover *fp-cop-F* can yield an unstable allocation with just these two hypotheses. Ergo we need to assume *irc* even when we have *lad*, unlike before (see §5.6).

This theorem is the foundation for all later strategic results.

```
locale ContractsWithUnilateralSubstitutesAndIRCAndLAD = ContractsWithUnilateralSubstitutesAndIRC + ContractsWithLAD

sublocale ContractsWithSubstitutesAndLAD < ContractsWithUnilateralSubstitutesAndIRCAndLAD

context ContractsWithUnilateralSubstitutesAndIRCAndLAD
begin

context
  fixes ds :: 'b set
  fixes X :: 'a set
```

```
  assumes stable-on ds X
begin
```

The proofs of these first two lemmas are provided by Hatfield and Kojima (2010, Theorem 6). We treat unemployment in the definition of the function A as we did in §5.1.3.

```
lemma RHT-Cd-card:
  assumes d ∈ ds
  shows card (Cd d X) ≤ card (Cd d (cop ds))

lemma RHT-Ch-card:
  shows card (Ch h (fp-cop-F ds)) ≤ card (Ch h X)
proof –
  define A where A ≡ λX. {y |y. Xd y ∈ ds ∧ y ∈ Field (Pd (Xd y)) ∧ (∀ x ∈ X. Xd x = Xd y ⟶ (x, y) ∈
Pd (Xd x))}
  have A (cop ds) = fp-cop-F ds (is ?lhs = ?rhs)
  proof(rule set-elem-equalityI)
    fix x assume x ∈ ?lhs
    show x ∈ ?rhs
    proof(cases Xd x ∈ Xd ‘ cop ds)
      case True with ‹x ∈ ?lhs› show ?thesis
        unfolding A-def by clarsimp (metis CH-range′ above-def fp-cop-F-closed-inv′ mem-Collect-eq)
    next
      case False with ‹x ∈ ?lhs› fp-cop-F-all show ?thesis
        unfolding A-def by blast
    qed
  next
    fix x assume x ∈ ?rhs
    with fp-cop-F-worst show x ∈ ?lhs
      unfolding A-def using fp-cop-F-range-inv′[OF ‹x ∈ ?rhs›] by fastforce
  qed
  moreover
  have CH (A X) = X
  proof(rule ccontr)
    assume CH (A X) ≠ X
    then have CH (A X) ≠ CH X using ‹stable-on ds X› stable-on-CH by blast
    then obtain h where XXX: Ch h (A X) ≠ Ch h X using mem-CH-Ch by blast
    have ¬stable-on ds X
    proof(rule blocking-on-imp-not-stable[OF blocking-onI])
      show Ch h (A X) ≠ Ch h X by fact
      from Pd-linear ‹stable-on ds X› show Ch h (A X) = Ch h (X ∪ Ch h (A X))
        unfolding A-def
        by – (rule consistencyD[OF Ch-consistency],
              auto 10 0 dest: Ch-range′ stable-on-Xd stable-on-range′ stable-on-allocation inj-onD underS-incl-iff)
    next
      fix x assume x ∈ Ch h (A X)
      with Ch-singular Pd-linear show x ∈ CD-on ds (X ∪ Ch h (A X))
        unfolding A-def
        by (auto 9 3 simp: mem-CD-on-Cd Cd-greatest greatest-def
                     dest: Ch-range′ Pd-range′ Cd-Xd Cd-single inj-onD underS-incl-iff
                    intro: FieldI1)
    qed
    with ‹stable-on ds X› show False by blast
  qed
  moreover
  from Pd-linear Theorem-5[OF ‹stable-on ds X›] ‹stable-on ds X› have A (cop ds) ⊆ A X
    unfolding A-def order-on-defs by (fastforce dest: Pd-Xd elim: transE)
  then have card (Ch h (A (cop ds))) ≤ card (Ch h (A X))
    by (fastforce intro: ladD[OF spec[OF Ch-lad]])
```

```
  ultimately show ?thesis by (metis (no-types, lifting) Ch-CH-irc-idem)
qed
```

The top-level proof is the same as in §5.6.

```
lemma Theorem-6-fp-cop-F:
  shows d ∈ ds ⟹ card (Cd d X) = card (Cd d (cop ds))
    and card (Ch h X) = card (Ch h (fp-cop-F ds))
proof -
  let ?Sum-Cd-COP = ∑ d∈ds. card (Cd d (cop ds))
  let ?Sum-Ch-COP = ∑ h∈UNIV. card (Ch h (fp-cop-F ds))
  let ?Sum-Cd-X = ∑ d∈ds. card (Cd d X)
  let ?Sum-Ch-X = ∑ h∈UNIV. card (Ch h X)

  have ?Sum-Cd-COP = ?Sum-Ch-COP
    using Theorem-1 stable-on-CD-on CD-on-card[symmetric] CH-card[symmetric] by simp
  also have ... ≤ ?Sum-Ch-X
    using RHT-Ch-card by (simp add: sum-mono)
  also have ... = ?Sum-Cd-X
    using CD-on-card[symmetric] CH-card[symmetric]
    using ‹stable-on ds X› stable-on-CD-on stable-on-CH by auto
  finally have ?Sum-Cd-X = ?Sum-Cd-COP
    using RHT-Cd-card by (simp add: eq-iff sum-mono)
  with RHT-Cd-card show d ∈ ds ⟹ card (Cd d X) = card (Cd d (cop ds))
    by (fastforce elim: sum-mono-inv)

  have ?Sum-Ch-X = ?Sum-Cd-X
    using ‹stable-on ds X› stable-on-CD-on stable-on-CH CD-on-card[symmetric] CH-card[symmetric] by simp
  also have ... ≤ ?Sum-Cd-COP
    using RHT-Cd-card by (simp add: sum-mono)
  also have ... = ?Sum-Ch-COP
    using CD-on-card[symmetric] CH-card[symmetric]
    using Theorem-1 stable-on-CD-on stable-on-CH by auto
  finally have ?Sum-Ch-COP = ?Sum-Ch-X
    using RHT-Ch-card by (simp add: eq-iff sum-mono)
  with RHT-Ch-card show card (Ch h X) = card (Ch h (fp-cop-F ds))
    by (fastforce elim: sym[OF sum-mono-inv])
qed

end

theorem Theorem-6:
  assumes stable-on ds X
  assumes stable-on ds Y
  shows d ∈ ds ⟹ card (Cd d X) = card (Cd d Y)
    and card (Ch h X) = card (Ch h Y)

end
```

6.5 Concluding remarks

We next discuss a kind of interference between doctors termed *bossiness* in §7. This has some implications for the strategic issues we discuss in §8.

7 Kojima (2010): The non-existence of a stable and non-bossy mechanism

Kojima (2010) says that "a mechanism is *nonbossy* if an agent cannot change [the] allocation of other agents unless doing so also changes her own allocation." He shows that no mechanism can be both *stable-on* and *nonbossy* in a one-to-one marriage market. We establish this result in our matching-with-contracts setting here.

There are two complications. Firstly, as not all agent preferences yield stable matches (unlike the marriage market), we constrain hospital choice functions to satisfy *ContractsWithBilateralSubstitutesAndIRC*, which is the weakest condition formalized here that ensures that *fp-cop-F* yields stable matches. (We note that it is not the weakest condition guaranteeing the existence of stable matches.)

Secondly, non-bossiness needs to separately treat the preferences of the doctors and the choice functions of the hospitals.

We work in the *Contracts* locale for its types and the constants *Xd* and *Xh*. To account for the quantification over preferences, we directly use some raw constants from the *Contracts* locale.

context *Contracts*
begin

abbreviation (*input*) *mechanism-domain* :: ($'d \Rightarrow 'x$ *rel*) $\Rightarrow$ ($'h \Rightarrow 'x$ *cfun*) $\Rightarrow$ *bool* **where**
mechanism-domain $\equiv$ *ContractsWithBilateralSubstitutesAndIRC Xd Xh*

definition *nonbossy* :: $'d$ *set* $\Rightarrow$ ($'d$, $'h$, $'x$) *mechanism* $\Rightarrow$ *bool* **where**
nonbossy ds $\varphi \longleftrightarrow$
$(\forall Pd\ Pd'\ Ch.\ \forall d{\in}ds.\ mechanism\text{-}domain\ Pd\ Ch \land mechanism\text{-}domain\ (Pd(d{:=}Pd'))\ Ch \longrightarrow$
$dX\ (\varphi\ Pd\ Ch\ ds)\ d = dX\ (\varphi\ (Pd(d{:=}Pd'))\ Ch\ ds)\ d \longrightarrow \varphi\ Pd\ Ch\ ds = \varphi\ (Pd(d{:=}Pd'))\ Ch\ ds)$
$\land\ (\forall Pd\ Ch\ Ch'\ h.\ mechanism\text{-}domain\ Pd\ Ch \land mechanism\text{-}domain\ Pd\ (Ch(h{:=}Ch')) \longrightarrow$
$hX\ (\varphi\ Pd\ Ch\ ds)\ h = hX\ (\varphi\ Pd\ (Ch(h{:=}Ch'))\ ds)\ h \longrightarrow \varphi\ Pd\ Ch\ ds = \varphi\ Pd\ (Ch(h{:=}Ch'))\ ds)$

definition *mechanism-stable* :: $'d$ *set* $\Rightarrow$ ($'d$, $'h$, $'x$) *mechanism* $\Rightarrow$ *bool* **where**
mechanism-stable ds φ
$\longleftrightarrow (\forall Pd\ Ch.\ mechanism\text{-}domain\ Pd\ Ch \longrightarrow Contracts.stable\text{-}on\ Pd\ Ch\ ds\ (\varphi\ Pd\ Ch\ ds))$

end

The proof is somewhat similar to those for Roth's impossibility results (see, for instance, Roth and Sotomayor (1990, Theorem 4.4)). It relies on the existence of at least three doctors, three hospitals, and a complete set of contracts between these. The following locale captures a suitable set of constraints.

locale *BossyConstants* =
fixes *Xd* :: $'x \Rightarrow 'd$
fixes *Xh* :: $'x \Rightarrow 'h$
fixes *d1h1 d1h2 d1h3* :: $'x$
fixes *d2h1 d2h2 d2h3* :: $'x$
fixes *d3h1 d3h2 d3h3* :: $'x$
fixes *ds* :: $'d$ *set*
assumes *ds*: *distinct* [*Xd d1h1*, *Xd d2h1*, *Xd d3h1*]
assumes *hs*: *distinct* [*Xh d1h1*, *Xh d1h2*, *Xh d1h3*]
assumes *Xd-xs*:
Xd ` $\{d1h2, d1h3\} = \{Xd\ d1h1\}$
Xd ` $\{d2h2, d2h3\} = \{Xd\ d2h1\}$
Xd ` $\{d3h2, d3h3\} = \{Xd\ d3h1\}$
assumes *Xh-xs*:
Xh ` $\{d2h1, d3h1\} = \{Xh\ d1h1\}$
Xh ` $\{d2h2, d3h2\} = \{Xh\ d1h2\}$
Xh ` $\{d2h3, d3h3\} = \{Xh\ d1h3\}$
assumes *dset*: $\{Xd\ d1h1, Xd\ d2h1, Xd\ d3h1\} \subseteq ds$

locale *ContractsWithBossyConstants* =
Contracts + *BossyConstants*
begin

abbreviation (*input*) *d1* $\equiv$ *Xd d1h1*
abbreviation (*input*) *d2* $\equiv$ *Xd d2h1*
abbreviation (*input*) *d3* $\equiv$ *Xd d3h1*

abbreviation (*input*) *h1* ≡ *Xh d1h1*
abbreviation (*input*) *h2* ≡ *Xh d1h2*
abbreviation (*input*) *h3* ≡ *Xh d1h3*

We proceed to show that variations on the following preferences for doctors and hospitals force a stable mechanism to be bossy. Recall that *linord-of-list* constructs a linear order from a list of elements greatest to least. The hospital choice functions take at most one contract from those on offer, and are again ordered from most preferable to least.

definition *BPd* :: *'b* ⇒ *'a rel* **where**
  BPd ≡ *map-of-default* {} [(*d1*, *linord-of-list* [*d1h3*, *d1h2*, *d1h1*])
                                , (*d2*, *linord-of-list* [*d2h3*, *d2h2*, *d2h1*])
                                , (*d3*, *linord-of-list* [*d3h1*, *d3h2*, *d3h3*])]

abbreviation *mkhord* :: *'d list* ⇒ *'d cfun* **where**
  mkhord xs X ≡ *set-option* (*List.find* (λ*x*. *x*∈*X*) *xs*)

definition *BCh* :: *'c* ⇒ *'a cfun* **where**
  BCh ≡ *map-of-default* (λ-. {}) [(*h1*, *mkhord* [*d1h1*, *d2h1*, *d3h1*])
                                    , (*h2*, *mkhord* [])
                                    , (*h3*, *mkhord* [*d3h3*, *d2h3*, *d1h3*])]

Interpreting the *Contracts* locale gives us access to some useful constants.

interpretation *Bossy*: *Contracts Xd Xh BPd BCh*

lemma *BPd-BCh-mechanism-domain*:
  shows *mechanism-domain BPd BCh*

lemma *Bossy-stable*:
  shows *Bossy.stable-on ds X* ⟷ *X* = {*d1h1*, *d3h3*}

The second preference order has doctor *d2* reject all contracts and is otherwise the same as the first.

definition *BPd'* :: *'b* ⇒ *'a rel* **where**
  BPd' = *BPd*(*d2* := {})

interpretation *Bossy'*: *Contracts Xd Xh BPd' BCh*

lemma *BPd'-BCh-mechanism-domain*:
  shows *mechanism-domain BPd' BCh*

lemma *Bossy'-stable*:
  shows *Bossy'.stable-on ds X* ⟷ *X* = {*d1h3*, *d3h1*} ∨ *X* = {*d1h1*, *d3h3*}

The third preference order adjusts the choice function of hospital *h2* and is otherwise the same as the second.

definition *BCh'* :: *'c* ⇒ *'a cfun* **where**
  BCh' ≡ *BCh*(*h2* := *mkhord* [*d1h2*, *d2h2*, *d3h2*])

interpretation *Bossy''*: *Contracts Xd Xh BPd' BCh'*

lemma *BPd'-BCh'-mechanism-domain*:
  shows *mechanism-domain BPd' BCh'*

lemma *Bossy''-stable*:
  shows *Bossy''.stable-on ds X* ⟷ *X* = {*d3h1*, *d1h3*}

theorem *Theorem-1*:
  shows ¬(*mechanism-stable ds* φ ∧ *nonbossy ds* φ)
proof(*rule notI*, *erule conjE*)

```
  assume S: Bossy.mechanism-stable ds φ
  assume NB: Bossy.nonbossy ds φ
  from S Bossy′-stable BPd′-BCh-mechanism-domain
  consider (A) φ BPd′ BCh ds = {d1h3, d3h1} | (B) φ BPd′ BCh ds = {d1h1, d3h3}
    unfolding mechanism-stable-def by blast
  then show False
  proof cases
    case A
    from S BPd-BCh-mechanism-domain Bossy-stable have φ BPd BCh ds = {d1h1, d3h3}
      unfolding mechanism-stable-def by blast
    with Xd-xs ds xs dset A show False
      using nonbossy-Pd[OF NB BPd-BCh-mechanism-domain BPd′-BCh-mechanism-domain[unfolded BPd′-def]]
      unfolding BPd′-def[symmetric] dX-def by fastforce
  next
    case B
    from S BPd′-BCh′-mechanism-domain Bossy″-stable have φ BPd′ BCh′ ds = {d3h1, d1h3}
      unfolding mechanism-stable-def by blast
    with Xh-xs hs xs dset B show False
     using nonbossy-Ch[OF NB BPd′-BCh-mechanism-domain BPd′-BCh′-mechanism-domain[unfolded BCh′-def]]
      unfolding BCh′-def[symmetric] hX-def by fastforce
  qed
qed
```

In particular, the COP (see §6) is bossy as it always yields stable matches under *mechanism-stable*.

```
theorem Theorem-1-COP:
  ¬nonbossy ds Contracts.cop
using ContractsWithBilateralSubstitutesAndIRC.Theorem-1 Theorem-1 mechanism-stable-def by blast

end
```

Therefore doctors can interfere with other doctors' allocations under the COP without necessarily disadvantaging themselves, which has implications for the notion of *group strategy-proof* (Hatfield and Kojima 2009); see §8.2.

8 Strategic results

We proceed to establish a series of strategic results for the COP (see §5.7 and §6), making use of the invariants we developed for it. These results also apply to the matching-with-contracts setting of §5, and where possible we specialize our lemmas to it.

8.1 Hatfield and Milgrom (2005): Theorems 10 and 11: Truthful revelation as a Dominant Strategy

Theorems 10 and 11 demonstrate that doctors cannot obtain better results for themselves in the doctor-optimal match (i.e., *cop ds*, equal to *match* (*gfp-F ds*) by *Theorem-15-match* assuming hospital preferences satisfy *substitutes*) by misreporting their preferences. (See Roth and Sotomayor (1990, §4.2) for a discussion about the impossibility of a mechanism being strategy-proof for all agents.)

Hatfield and Milgrom (2005, §III(B)) provide the following intuition:

> We will show the positive incentive result for the doctor-offering algorithm in two steps which highlight the different roles of the two preference assumptions. First, we show that the *substitutes* condition, by itself, guarantees that doctors cannot benefit by exaggerating the ranking of an unattainable contract. More precisely, if there exists a preferences list for a doctor d such that d obtains contract x by submitting this list, then d can also obtain x by submitting a preference list that includes only contract x [Theorem 10]. Second, we will show that adding the law of aggregate demand guarantees that a doctor does at least as well as reporting truthfully as by reporting any singleton [Theorem 11]. Together, these are the dominant strategy result.

We prove Theorem 10 via a lemma that states that the contracts above $x \in X$ for some stable match X with respect to manipulated preferences $Pd\ (Xd\ x)$ do not improve the outcome for doctor $Xd\ x$ with respect to their true preferences $Pd'\ (Xd\ x)$ in the doctor-optimal match for Pd'.

This is weaker than Hatfield and Kojima (2009, Lemma 1) (see §8.2) as we do not guarantee that the allocation does not change. By the bossiness result of §7, such manipulations can change the outcomes of the other doctors; this lemma establishes that only weak improvements are possible.

```
context ContractsWithUnilateralSubstitutesAndIRC
begin

context
  fixes d' :: 'b
  fixes Pd' :: 'b ⇒ 'a rel
  assumes Pd'-d'-linear: Linear-order (Pd' d')
  assumes Pd'-d'-range: Field (Pd' d') ⊆ {y. Xd y = d'}
  assumes Pd': ∀ d. d≠d' ⟶ Pd' d = Pd d
begin

interpretation PdXXX: ContractsWithUnilateralSubstitutesAndIRC Xd Xh Pd' Ch

theorem Pd-above-irrelevant:
  assumes d'-Field: dX X d' ⊆ Field (Pd' d')
  assumes d'-Above: Above (Pd' d') (dX X d') ⊆ Above (Pd d') (dX X d')
  assumes x ∈ X
  assumes stable-on ds X
  shows ∃ y ∈ PdXXX.cop ds. (x, y) ∈ Pd' (Xd x)
proof(rule PdXXX.Theorem-5[OF ccontr ‹x ∈ X›])
  assume ¬PdXXX.stable-on ds X
  then show False
  proof(cases rule: PdXXX.not-stable-on-cases)
    case not-individually-rational
    from Pd' ‹stable-on ds X› d'-Field have x ∈ PdXXX.Cd (Xd x) X if x ∈ X for x
      using that unfolding dX-def by (force simp: stable-on-range' stable-on-allocation PdXXX.Cd-single)
    with ‹stable-on ds X› not-individually-rational show False
      unfolding PdXXX.individually-rational-on-def
      by (auto simp: PdXXX.mem-CD-on-Cd stable-on-Xd dest: stable-on-CH PdXXX.CD-on-range')
  next
    case not-no-blocking
    then obtain h X'' where PdXXX.blocking-on ds X h X''
      unfolding PdXXX.stable-no-blocking-on-def by blast
    have blocking-on ds X h X''
    proof(rule blocking-onI)
      fix x assume x ∈ X''
      note Pbos = PdXXX.blocking-on-Field[OF ‹PdXXX.blocking-on ds X h X''›]
                  PdXXX.blocking-on-allocation[OF ‹PdXXX.blocking-on ds X h X''›]
                  PdXXX.blocking-on-CD-on'[OF ‹PdXXX.blocking-on ds X h X''› ‹x ∈ X''›]
      show x ∈ CD-on ds (X ∪ X'')
      proof(cases Xd x = d')
        case True
        from Pd-linear' d'-Field d'-Above ‹x ∈ X''› ‹Xd x = d'› Pbos
        have dX X'' (Xd x) ⊆ Field (Pd (Xd x))
          by (force simp: PdXXX.mem-CD-on-Cd PdXXX.Cd-Above PdXXX.dX-Int-Field-Pd Above-union
                        Int-Un-distrib2 dX-singular intro: Above-Field)
        moreover from ‹stable-on ds X› have dX X (Xd x) ⊆ Field (Pd (Xd x))
          by (force dest: dX-range' stable-on-range')
        moreover note Pd-linear' Pd-range PdXXX-range d'-Field d'-Above ‹x ∈ X''› ‹Xd x = d'› Pbos
        ultimately show ?thesis
          by (clarsimp simp: PdXXX.mem-CD-on-Cd PdXXX.Cd-Above-dX mem-CD-on-Cd Cd-Above-dX
```

Above-union dX-union Int-Un-distrib2)
(*fastforce simp*: *dX-singular intro*: *Above-Linear-singleton*)
next
case *False*
with ‹$x \in$ *PdXXX.CD-on ds* $(X \cup X'')$› **show** *?thesis*
by (*clarsimp simp*: *Pd′ PdXXX.mem-CD-on-Cd mem-CD-on-Cd PdXXX.Cd-greatest Cd-greatest*)
qed
qed (*use* ‹*PdXXX.blocking-on ds X h X′′*› **in** ‹*simp-all add*: *PdXXX.blocking-on-def*›)
with ‹*stable-on ds X*› **show** *False* **by** (*simp add*: *blocking-on-imp-not-stable*)
qed
qed

end

end

We now specialize this lemma to Theorem 10 by defining a preference order for the doctors where distinguished doctors *ds* submit single preferences for the contracts they receive in the doctor-optimal match.

The function *override-on f g A* = ($\lambda a.$ *if* $a \in A$ *then g a else f a*) denotes function update at several points.

context *Contracts*
begin

definition *Pd-singletons-for-ds* :: *′x set* ⇒ *′d set* ⇒ *′d* ⇒ *′x rel* **where**
Pd-singletons-for-ds X ds ≡ *override-on Pd* ($\lambda d.$ *dX X d* × *dX X d*) *ds*

end

We interpret our *ContractsWithUnilateralSubstitutesAndIRC* locale with respect to this updated preference order, which gives us the stable match and properties of it.

context *ContractsWithUnilateralSubstitutesAndIRC*
begin

context
fixes *ds* :: *′b set*
fixes *X* :: *′a set*
assumes *stable-on ds X*
begin

interpretation
Singleton-for-d: *ContractsWithUnilateralSubstitutesAndIRC Xd Xh Pd-singletons-for-ds X* {*d*} *Ch* **for** *d*

Our version of Hatfield and Milgrom (2005, Theorem 10) (for the COP) states that if a doctor submits a preference order containing just *x*, where *x* is their contract in some stable match *X*, then that doctor receives exactly *x* in the doctor-optimal match and all other doctors do at least as well.

theorem *Theorem-10-fp-cop-F*:
assumes $x \in X$
shows ∃ *y* ∈ *Singleton-for-d.cop d ds*. (*x*, *y*) ∈ *Pd-singletons-for-ds X* {*d*} (*Xd x*)
proof(*rule Pd-above-irrelevant*[**where** *ds*=*ds* **and** *d′*=*d* **and** *X*=*X*])
from *stable-on-allocation* ‹*stable-on ds X*›
show *Above* (*Pd-singletons-for-ds X* {*d*} *d*) (*Singleton-for-d.dX X d*) ⊆ *Above* (*Pd d*) (*Singleton-for-d.dX X d*)
by (*clarsimp simp*: *Above-def Pd-singletons-for-ds-simps dX-def*) (*metis inj-on-eq-iff stable-on-range′ Pd-refl*)
qed (*use stable-on-allocation* ‹*stable-on ds X*› *Pd-singletons-for-ds-linear Pd-singletons-for-ds-range assms*
in ‹*simp-all*, *simp-all add*: *Pd-singletons-for-ds-simps dX-def*›)

end

end

We can recover the original Theorem 10 by specializing this result to *gfp-F*.

context *ContractsWithSubstitutesAndIRC*
begin

interpretation
Singleton-for-d: *ContractsWithSubstitutesAndIRC Xd Xh Pd-singletons-for-ds* (*match* (*gfp-F ds*)) {*d*} *Ch*
for *ds d*

theorem *Theorem-10*:
assumes $x \in$ *match* (*gfp-F ds*)
shows $\exists\, y \in$ *match* (*Singleton-for-d.gfp-F ds d ds*). $(x, y) \in$ *Pd-singletons-for-ds* (*match* (*gfp-F ds*)) {*d*} (*Xd x*)
using *Theorem-10-fp-cop-F Singleton-for-d.Theorem-15-match Theorem-15-match gfp-F-stable-on assms* **by** *simp*

corollary *Theorem-10-d*:
assumes $x \in$ *match* (*gfp-F ds*)
shows $x \in$ *match* (*Singleton-for-d.gfp-F ds* (*Xd x*) *ds*)
using *gfp-F-stable-on*[*of ds*] *Theorem-10*[*OF assms*(*1*), *of Xd x*] *assms*
by (*clarsimp simp*: *Pd-singletons-for-ds-simps dX-def inj-on-eq-iff dest*!: *stable-on-allocation*)

end

The second theorem (Hatfield and Milgrom 2005, Theorem 11) depends on both Theorem 10 and the rural hospitals theorem (§5.6, §6.4). It shows that, assuming everything else is fixed, if doctor d' obtains contract x with (manipulated) preferences $Pd\ d'$ in the doctor-optimal match, then they will obtain a contract at least as good by submitting their true preferences $Pd'\ d'$ (with respect to these true preferences).

locale *TruePrefs* = *Contracts* +
fixes x :: $'a$
fixes X :: $'a$ *set*
fixes *ds* :: $'b$ *set*
fixes Pd' :: $'b \Rightarrow {'a}$ *rel*
assumes x: $x \in X$
assumes X: *stable-on ds X*
assumes Pd'-d'-x: $x \in$ *Field* (Pd' (*Xd x*))
assumes Pd'-d'-*linear*: *Linear-order* (Pd' (*Xd x*))
assumes Pd'-d'-*range*: *Field* (Pd' (*Xd x*)) $\subseteq$ {*y*. *Xd y* = *Xd x*}
assumes Pd': $\forall\, d.\ d \neq Xd\ x \longrightarrow Pd'\ d = Pd\ d$

locale *ContractsWithUnilateralSubstitutesAndIRCAndLADAndTruePrefs* =
ContractsWithUnilateralSubstitutesAndIRCAndLAD + *TruePrefs*
begin

interpretation *TruePref*: *ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh* Pd' *Ch*

interpretation *TruePref-tax*: *ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh* Pd'-*tax Ch*

interpretation
Singleton-for-d: *ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh Pd-singletons-for-ds X* {*Xd x*} *Ch*

lemma *Theorem-11-*Pd'*-tax*:
shows $\exists\, y \in$ *TruePref-tax.cop ds*. $(x, y) \in Pd'$-*tax* (*Xd x*)
proof(*rule ccontr*)
let *?Z* = *TruePref-tax.cop ds*
assume $\neg$*?thesis* **then have** *Xd x* $\notin$ *Xd* ` *?Z*
using Pd'-*range* Pd'-*linear*[*of Xd x*] Pd'-d'-x **unfolding** *order-on-defs*
by − (*clarsimp*, *drule* (*1*) *bspec*,
fastforce simp: Pd'-*tax-def above-def Refl-Field-Restr dest*: *refl-onD*
dest!: *CH-range*′ *TruePref-tax.fp-cop-F-range-inv*′)
show *False*

```
proof(cases Singleton-for-d.stable-on ds ?Z)
  case True
  moreover
  from Theorem-10-fp-cop-F[OF X x, of Xd x] X
  have x ∈ CH (Singleton-for-d.fp-cop-F ds)
    by (force simp: Pd-singletons-for-ds-simps dX-def dest: inj-onD stable-on-allocation)
  with Singleton-for-d.fp-cop-F-allocation
  have Singleton-for-d.Cd (Xd x) (Singleton-for-d.cop ds) = {x}
    by (meson Singleton-for-d.Cd-single Singleton-for-d.Cd-singleton Singleton-for-d.fp-cop-F-range-inv′
              TruePref-tax.CH-range′)
  with Singleton-for-d.Theorem-1[of ds]
  have x ∈ Y if Singleton-for-d.stable-on ds Y for Y
    using Singleton-for-d.Theorem-6-fp-cop-F(1)[where ds=ds and X=Y and d=Xd x] that Xd-x-ds x
          card-Suc-eq[where A=Singleton-for-d.Cd (Xd x) Y and k=0] stable-on-allocation[OF X]
    by (fastforce simp: Singleton-for-d.Cd-singleton[symmetric] Pd-singletons-for-ds-simps dX-def
                  dest: Singleton-for-d.Cd-range′ inj-onD)
  moreover note ‹Xd x ∉ Xd ‘ ?Z›
  ultimately show False by blast
next
  case False note ‹¬Singleton-for-d.stable-on ds ?Z›
  then show False
  proof(cases rule: Singleton-for-d.not-stable-on-cases)
    case not-individually-rational
    with TruePref-tax.Theorem-1[of ds] ‹Xd x ∉ Xd ‘ ?Z›
    show False
      unfolding TruePref-tax.stable-on-def Singleton-for-d.individually-rational-on-def
                TruePref-tax.individually-rational-on-def Singleton-for-d.CD-on-def
      by (auto dest: Singleton-for-d.Cd-range′)
         (metis TruePref-tax.mem-CD-on-Cd TruePref-tax-Cd-not-x image-eqI)
  next
    case not-no-blocking
    then obtain h X″ where Singleton-for-d.blocking-on ds ?Z h X″
      unfolding Singleton-for-d.stable-no-blocking-on-def by blast
    have TruePref-tax.blocking-on ds ?Z h X″
    proof(rule TruePref-tax.blocking-onI)
      fix y assume y ∈ X″
      with ‹Singleton-for-d.blocking-on ds ?Z h X″› have YYY: y ∈ Singleton-for-d.CD-on ds (?Z ∪ X″)
        unfolding Singleton-for-d.blocking-on-def by blast
      show y ∈ TruePref-tax.CD-on ds (?Z ∪ X″)
      proof(cases Xd y = Xd x)
        case True
        with inj-on-eq-iff[OF stable-on-allocation x] X YYY have y = x
          by (fastforce simp: Singleton-for-d.mem-CD-on-Cd Pd-singletons-for-ds-simps dX-def
                       dest: Singleton-for-d.Cd-range′)
        with X Xd-x-ds TruePref-tax.Theorem-1[of ds] ‹Xd x ∉ Xd ‘ ?Z› ‹y ∈ X″›
        show ?thesis
          using Singleton-for-d.blocking-on-allocation[OF ‹Singleton-for-d.blocking-on ds ?Z h X″›]
          by (clarsimp simp: TruePref-tax.mem-CD-on-Cd TruePref-tax.Cd-greatest greatest-def Pd′-tax-x)
             (metis TruePref-tax.Pd-range′ image-eqI inj-on-contraD TruePref-tax.Pd-refl)
      next
        case False with YYY show ?thesis
          by (simp add: Singleton-for-d.mem-CD-on-Cd TruePref-tax.mem-CD-on-Cd TruePref-tax-Cd-not-x)
      qed
    qed (use ‹Singleton-for-d.blocking-on ds ?Z h X″› in ‹simp-all add: Singleton-for-d.blocking-on-def›)
    with TruePref-tax.Theorem-1[of ds] show False by (simp add: TruePref-tax.blocking-on-imp-not-stable)
  qed
 qed
qed
```

```
theorem Theorem-11-fp-cop-F:
  shows ∃ y∈TruePref.cop ds. (x, y) ∈ Pd′ (Xd x)
proof −
  from Theorem-11-Pd′-tax
  obtain y where y: y ∈ CH (TruePref-tax.fp-cop-F ds)
            and xy: (x, y) ∈ Pd′-tax (Xd x) ..
  from TruePref-tax.stable-on-range′[OF TruePref-tax.Theorem-1]
  have dX (CH (TruePref-tax.fp-cop-F ds)) (Xd x) ⊆ Field (Pd′ (Xd x))
    by (clarsimp simp: dX-def) (metis (no-types, opaque-lifting) Pd′-tax-Pd′ contra-subsetD mono-Field)
  moreover
  from TruePref-tax.fp-cop-F-allocation[of ds] Pd′-tax-Pd′ y xy
  have Above (Pd′ (Xd x)) (dX (CH (TruePref-tax.fp-cop-F ds)) (Xd x))
    ⊆ Above (Pd′-tax (Xd x)) (dX (CH (TruePref-tax.fp-cop-F ds)) (Xd x))
    by − (rule Pd′-Above; fastforce simp: dX-singular above-def dest: TruePref-tax.Pd-Xd)
  moreover note Pd′-linear Pd′-range TruePref-tax.Theorem-1[of ds] y
  ultimately have z: ∃ z∈CH (TruePref.fp-cop-F ds). (y, z) ∈ Pd′ (Xd y)
    by − (rule TruePref-tax.Pd-above-irrelevant[where d′=Xd x and X=CH (TruePref-tax.fp-cop-F ds)];
         simp add: Pd′-tax-def)
  from Pd′-linear xy z show ?thesis
    unfolding Pd′-tax-def order-on-defs by clarsimp (metis TruePref.Pd-Xd transE)
qed

end

locale ContractsWithSubstitutesAndLADAndTruePrefs =
  ContractsWithSubstitutesAndLAD + TruePrefs

sublocale ContractsWithSubstitutesAndLADAndTruePrefs
        < ContractsWithUnilateralSubstitutesAndIRCAndLADAndTruePrefs

context ContractsWithSubstitutesAndLADAndTruePrefs
begin

interpretation TruePref: ContractsWithSubstitutesAndLAD Xd Xh Pd′ Ch

theorem Theorem-11:
  shows ∃ y∈match (TruePref.gfp-F ds). (x, y) ∈ Pd′ (Xd x)
using Theorem-11-fp-cop-F TruePref.Theorem-15-match by simp

end
```

Note that this theorem depends on the hypotheses introduced by the *TruePrefs* locale, and only applies to doctor *Xd x*. The following sections show more general and syntactically self-contained results.

We omit Hatfield and Milgrom (2005, Theorem 12), which demonstrates the almost-necessity of LAD for truth revelation to be the dominant strategy for doctors.

8.2 Hatfield and Kojima (2009, 2010): The doctor-optimal match is group strategy-proof

Hatfield and Kojima (2010, Theorem 7) assert that the COP is group strategy-proof, which we define below. We begin by focusing on a single agent (Hatfield and Kojima 2009):

> A mechanism φ is *strategy-proof* if, for any preference profile Pd, there is no doctor d and preferences Pd' such that d strictly prefers y_d to x_d according to $Pd\ d$, where x_d and y_d are the (possibly null) contracts for d in $\varphi\ Pd$ and $\varphi\ Pd(d := Pd')$, respectively.

The syntax $f(a := b) = (\lambda x.\ if\ x = a\ then\ b\ else\ f\ x)$ denotes function update at a point.

We make this definition in the *Contracts* locale to avail ourselves of some types and the *Xd* and *Xh* constants. We also restrict hospital preferences to those that guarantee our earlier strategic results. As *gfp-F* requires these to satisfy the stronger *substitutes* constraint for stable matches to exist, we now deal purely with the COP.

context *Contracts*
begin

abbreviation (*input*) *mechanism-domain* :: $('d \Rightarrow 'x\ rel) \Rightarrow ('h \Rightarrow 'x\ cfun) \Rightarrow bool$ **where**
mechanism-domain $\equiv$ *ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh*

definition *strategy-proof* :: $'d\ set \Rightarrow ('d,\ 'h,\ 'x)\ mechanism \Rightarrow bool$ **where**
strategy-proof ds $\varphi \longleftrightarrow$
$(\forall\, Pd\ Ch.\ mechanism\text{-}domain\ Pd\ Ch \longrightarrow$
$\neg(\exists\, d{\in}ds.\ \exists\, Pd'.\ mechanism\text{-}domain\ (Pd(d{:=}Pd'))\ Ch$
$\wedge\ (\exists\, y{\in}\varphi\ (Pd(d{:=}Pd'))\ Ch\ ds.\ y \in AboveS\ (Pd\ d)\ (dX\ (\varphi\ Pd\ Ch\ ds)\ d))))$

theorem *fp-cop-F-strategy-proof*:
shows *strategy-proof ds Contracts.cop* (**is** *strategy-proof - ?*φ)

end

The adaptation to groups is straightforward (Hatfield and Kojima 2009, 2010):

> A mechanism φ is *group strategy-proof* if, for any preference profile *Pd*, there is no group of doctors $ds' \subseteq ds$ and a preference profile Pd' such that every $d \in ds'$ strictly prefers y_d to x_d according to *Pd d*, where x_d and y_d are the (possibly null) contracts for *d* in φ *Pd* and $\varphi\ Pd(d_1 := Pd'\ d_1, \ldots, d_n := Pd'\ d_n)$, respectively.

This definition requires all doctors in the coalition to strictly prefer the outcome with manipulated preferences, as Kojima's bossiness results (see §7) show that a doctor may influence other doctors' allocations without affecting their own. See Hatfield and Kojima (2009, §3) for discussion, and also Roth and Sotomayor (1990, Chapter 4); in particular their §4.3.1 discusses the robustness of these results and exogenous transfers.

context *Contracts*
begin

definition *group-strategy-proof* :: $'d\ set \Rightarrow ('d,\ 'h,\ 'x)\ mechanism \Rightarrow bool$ **where**
group-strategy-proof ds $\varphi \longleftrightarrow$
$(\forall\, Pd\ Ch.\ mechanism\text{-}domain\ Pd\ Ch \longrightarrow$
$\neg(\exists\, ds'{\subseteq}ds.\ ds' \neq \{\} \wedge (\exists\, Pd'.\ mechanism\text{-}domain\ (override\text{-}on\ Pd\ Pd'\ ds')\ Ch$
$\wedge\ (\forall\, d{\in}ds'.\ \exists\, y{\in}\varphi\ (override\text{-}on\ Pd\ Pd'\ ds')\ Ch\ ds.\ y \in AboveS\ (Pd\ d)\ (dX\ (\varphi\ Pd\ Ch\ ds)\ d)))))$

lemma *group-strategy-proof-strategy-proof*:
assumes *group-strategy-proof ds* φ
shows *strategy-proof ds* φ

end

Perhaps surprisingly, Hatfield and Kojima (2010, Lemma 1, for a single doctor) assert that shuffling any contract above the doctor-optimal one to the top of a doctor's preference order preserves exactly the doctor-optimal match, which on the face of it seems to contradict the bossiness result of §7: by the earlier strategy-proofness results, this cannot affect the outcome for that particular doctor, but by bossiness it may affect others. The key observation is that this manipulation preserves blocking coalitions in the presence of *lad*.
This result is central to showing the group-strategy-proofness of the COP.

context *Contracts*
begin

definition *shuffle-to-top* :: $'x\ set \Rightarrow 'd \Rightarrow 'x\ rel$ **where**
shuffle-to-top $Y = (\lambda d.\ Pd\ d - dX\ Y\ d \times UNIV \cup (Domain\ (Pd\ d) \cup dX\ Y\ d) \times dX\ Y\ d)$

```
definition Pd-shuffle-to-top :: 'd set ⇒ 'x set ⇒ 'd ⇒ 'x rel where
  Pd-shuffle-to-top ds' Y = override-on Pd (shuffle-to-top Y) ds'

end

context ContractsWithUnilateralSubstitutesAndIRCAndLAD
begin

lemma Lemma-1:
  assumes allocation Y
  assumes III: ∀ d∈ds''. ∃ y∈Y. y ∈ AboveS (Pd d) (dX (cop ds) d)
  shows cop ds = Contracts.cop (Pd-shuffle-to-top ds'' Y) Ch ds
using finite[of ds''] subset-refl
proof(induct ds'' rule: finite-subset-induct')
  case empty show ?case by (simp add: Pd-shuffle-to-top-simps)
next
  case (insert d ds')
  from insert
  interpret Pds': ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh Pd-shuffle-to-top ds' Y Ch
  let ?Z = CH (Pds'.fp-cop-F ds)
  note IH = ‹cop ds = ?Z›
  let ?Pd-shuffle-to-top = Pd-shuffle-to-top (insert d ds') Y
  from insert interpret Pdds': ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh ?Pd-shuffle-to-top Ch
  have XXX: ?Z = CH (Pdds'.fp-cop-F ds)
 proof(rule Pdds'.doctor-optimal-match-unique[OF Pdds'.doctor-optimal-matchI Pdds'.fp-cop-F-doctor-optimal-match])
    show Pdds'.stable-on ds ?Z
    proof(rule Pdds'.stable-onI)
      show Pdds'.individually-rational-on ds ?Z
      proof(rule Pdds'.individually-rational-onI)
        show Pdds'.CD-on ds ?Z = ?Z (is ?lhs = ?rhs)
        proof(rule set-elem-equalityI)
          fix x assume x ∈ ?rhs
          with ‹allocation Y› IH Pds'.Theorem-1[of ds] ‹d ∉ ds'› show x ∈ ?lhs
            by (clarsimp simp: Pds'.stable-on-Xd Pdds'.mem-CD-on-Cd Pdds'.Cd-greatest greatest-def
                          Pd-shuffle-to-top-Field[OF ‹allocation Y›],
                simp add: Pd-shuffle-to-top-simps shuffle-to-top-def dX-def Set.Ball-def,
                metis stable-on-range'[OF Theorem-1[of ds]] inj-on-contraD[OF Pds'.fp-cop-F-allocation[of ds]]
                    fp-cop-F-worst[of - ds] Pd-range' Pds'.CH-range')
        qed (meson IntE Pdds'.CD-on-range')
        show CH ?Z = ?Z by (simp add: CH-irc-idem)
      qed
      show Pdds'.stable-no-blocking-on ds ?Z
      proof(rule Pdds'.stable-no-blocking-onI2)
        fix h X'' assume Pbo: Pdds'.blocking-on ds ?Z h X''
        have Pds'.blocking-on ds ?Z h X''
        proof(rule Pds'.blocking-onI)
          fix x assume x ∈ X''
          note Pbos = Pdds'.blocking-on-allocation[OF ‹Pdds'.blocking-on ds ?Z h X''›]
                   Pdds'.blocking-on-CD-on'[OF ‹Pdds'.blocking-on ds ?Z h X''› ‹x ∈ X''›]
                   Pdds'.blocking-on-Cd[OF ‹Pdds'.blocking-on ds ?Z h X''›, where d=Xd x]
          show x ∈ Pds'.CD-on ds (?Z ∪ X'')
          proof(cases Xd x = d)
            case True
            from ‹allocation Y› III ‹d ∈ ds''› ‹Xd x = d›
            have dX Y (Xd x) ⊆ Field (Pd (Xd x))
                by clarsimp (metis AboveS-Pd-Xd AboveS-Field dX-range' inj-on-eq-iff)
            moreover with ‹allocation Y› ‹d ∉ ds'›
                      Pdds'.blocking-on-Field[OF ‹Pdds'.blocking-on ds ?Z h X''›, where d=d] ‹Xd x = d›
```

```
      have dX X'' (Xd x) ⊆ Field (Pd (Xd x))
        by (force simp: Pd-shuffle-to-top-simps shuffle-to-top-Field)
      moreover note ‹allocation Y› bspec[OF III[unfolded IH] ‹d ∈ ds''›] ‹d ∉ ds'› ‹x ∈ X''› ‹Xd x = d›
                    Pds'.stable-on-allocation[OF Pds'.Theorem-1] Pbos
      ultimately show ?thesis
        by (clarsimp simp: Pdds'.mem-CD-on-Cd Pds'.mem-CD-on-Cd Pds'.Cd-Above Pdds'.Cd-Above
                            Int-Un-distrib2 Pd-shuffle-to-top-Field)
           (clarsimp simp: Pd-shuffle-to-top-simps dX-singular dX-Int-Field-Pd;
            fastforce simp: Above-def AboveS-def Pd-refl shuffle-to-top-def dX-def intro: FieldI1 dest: Pd-range'
iff: inj-on-eq-iff)
     next
      case False
      from Pbos ‹Xd x ≠ d›
      show ?thesis
        by (simp add: Pdds'.mem-CD-on-Cd Pds'.mem-CD-on-Cd Pds'.Cd-greatest Pdds'.Cd-greatest)
           (simp add: Pd-shuffle-to-top-simps)
     qed
    qed (use ‹Pdds'.blocking-on ds ?Z h X''› in ‹simp-all add: Pdds'.blocking-on-def›)
    with Pds'.Theorem-1[of ds] show False by (simp add: Pds'.blocking-on-imp-not-stable)
   qed
  qed
 next
  fix W w assume Pdds'.stable-on ds W w ∈ W
  from III ‹d ∈ ds''› IH
  obtain y where Y: y ∈ Y y ∈ AboveS (Pd d) (dX (Pds'.cop ds) d) Xd y = d
    by (metis AboveS-Pd-Xd)
  show ∃ z∈Pds'.cop ds. (w, z) ∈ Pd-shuffle-to-top (insert d ds') Y (Xd w)
  proof(cases y ∈ W)
    case True note ‹y ∈ W›
    from ‹d ∉ ds'› ‹Pdds'.stable-on ds W› Y ‹y ∈ W›
    interpret Pdds': ContractsWithUnilateralSubstitutesAndIRCAndLADAndTruePrefs
                      Xd Xh Pd-shuffle-to-top (insert d ds') Y Ch y W ds Pd-shuffle-to-top ds' Y
    from ‹d ∉ ds'› Y Pdds'.Theorem-11-fp-cop-F have False
      using Pds'.stable-on-allocation[OF Pds'.Theorem-1[of ds]] Pd-linear Pd-range'
      unfolding order-on-defs antisym-def AboveS-def dX-def
      by (clarsimp simp: Pd-shuffle-to-top-simps) (blast dest: Pd-Xd)
    then show ?thesis ..
  next
    case False note ‹y ∉ W›
    show ?thesis
    proof (cases Pds'.stable-on ds W)
      case True note ‹Pds'.stable-on ds W›
      with ‹allocation Y› ‹d ∉ ds'› Y ‹w ∈ W› ‹y ∉ W› show ?thesis
        using Pds'.Theorem-5[OF ‹Pds'.stable-on ds W› ‹w ∈ W›]
        by (auto 0 2 simp: Pd-shuffle-to-top-simps shuffle-to-top-def dX-def AboveS-def dest: Pd-range' inj-onD)
    next
      case False note ‹¬Pds'.stable-on ds W›
      then show ?thesis
      proof(cases rule: Pds'.not-stable-on-cases)
        case not-individually-rational
        note Psos = Pdds'.stable-on-allocation[OF ‹Pdds'.stable-on ds W›]
                    Pdds'.stable-on-CH[OF ‹Pdds'.stable-on ds W›]
                    Pdds'.stable-on-Xd[OF ‹Pdds'.stable-on ds W›]
        have x ∈ Pds'.Cd (Xd x) W if x ∈ W for x
        proof(cases Xd x = d)
          case True
          with ‹allocation Y› ‹allocation W› Y(1,3) ‹y ∉ W›
               Pdds'.stable-on-range'[OF ‹Pdds'.stable-on ds W› ‹x ∈ W›] ‹x ∈ W›
```

```
          show ?thesis by (force simp: Pd-shuffle-to-top-Field dest: dX-range' inj-onD intro: Pds'.Cd-single)
        next
          case False
          with ‹allocation Y› ‹allocation W› Pdds'.stable-on-range'[OF ‹Pdds'.stable-on ds W› ‹x ∈ W›] ‹x ∈
W›
          show ?thesis by (auto simp: Pd-shuffle-to-top-Field intro!: Pds'.Cd-single)
        qed
        with not-individually-rational ‹Pdds'.CH W = W› Psos(3) show ?thesis
          unfolding Pds'.individually-rational-on-def by (auto simp: Pds'.mem-CD-on-Cd dest: Pds'.Cd-range')
      next
        case not-no-blocking
        then obtain h X'' where Pbo: Pds'.blocking-on ds W h X''
          unfolding Pds'.stable-no-blocking-on-def by blast
        have Pdds'.blocking-on ds W h X''
        proof(rule Pdds'.blocking-onI)
          fix x assume x ∈ X''
          note Pbos = Pds'.blocking-on-allocation[OF ‹Pds'.blocking-on ds W h X''›]
                      Pds'.blocking-on-CD-on'[OF ‹Pds'.blocking-on ds W h X''› ‹x ∈ X''›]
                      Pds'.blocking-on-Field[OF ‹Pds'.blocking-on ds W h X''›, where d=d]
          show x ∈ Pdds'.CD-on ds (W ∪ X'')
          proof(cases Xd x = d)
            case True
            from ‹allocation Y› III ‹d ∈ ds''›  ‹Xd x = d›
            have dX Y (Xd x) ⊆ Field (Pd (Xd x))
              by clarsimp (metis AboveS-Pd-Xd AboveS-Field dX-range' inj-on-eq-iff)
            moreover with ‹d ∉ ds'› ‹Xd x = d› Pbos
            have dX X'' (Xd x) ⊆ Field (Pd (Xd x))
              by (clarsimp simp: Pd-shuffle-to-top-simps)
            moreover note ‹allocation Y› ‹d ∉ ds'› ‹y ∉ W› ‹Xd y = d› ‹x ∈ X''› Pbos
            ultimately show ?thesis
              by (clarsimp simp: Pdds'.mem-CD-on-Cd Pds'.mem-CD-on-Cd Pds'.Cd-Above Pdds'.Cd-Above
                                 Int-Un-distrib2)
               (clarsimp simp: Pd-shuffle-to-top-simps shuffle-to-top-Field dX-singular dX-Int-Field-Pd Un-absorb2,
                  force simp: ‹y ∈ Y› shuffle-to-top-def dX-def Above-def dest: inj-onD intro: FieldI1)
          next
            case False
            from Pbos ‹Xd x ≠ d› show ?thesis
              by (simp add: Pdds'.mem-CD-on-Cd Pds'.mem-CD-on-Cd Pds'.Cd-greatest Pdds'.Cd-greatest)
                 (simp add: Pd-shuffle-to-top-simps)
          qed
        qed (use ‹Pds'.blocking-on ds W h X''› in ‹simp-all add: Pds'.blocking-on-def›)
        with ‹Pdds'.stable-on ds W› have False by (simp add: Pdds'.blocking-on-imp-not-stable)
        then show ?thesis ..
      qed
    qed
  qed
qed
from ‹?Z = CH (Pdds'.fp-cop-F ds)› IH show cop ds = Pdds'.cop ds by simp
qed
```

The top-level theorem states that the COP is group strategy proof. To account for the quantification over preferences, we directly use the raw constants from the *Contracts* locale.

```
theorem fp-cop-F-group-strategy-proof:
  shows group-strategy-proof ds Contracts.cop
        (is group-strategy-proof - ?φ)
proof(rule group-strategy-proofI)
  fix Pd Pds' Ch ds'
  assume XXX: mechanism-domain Pd Ch mechanism-domain (override-on Pd Pds' ds') Ch
```

```
  and YYY: ds′ ⊆ ds ds′ ≠ {}
  and ZZZ: ∀ d∈ds′. ∃ y∈?φ (override-on Pd Pds′ ds′) Ch ds. y ∈ AboveS (Pd d) (dX (?φ Pd Ch ds) d)
from XXX(1) interpret TruePref: ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh Pd Ch .
from XXX(2) interpret
  ManiPref: ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh override-on Pd Pds′ ds′ Ch .
let ?Y = ManiPref.cop ds
let ?Z = TruePref.cop ds
let ?Pd-shuffle-to-top = TruePref.Pd-shuffle-to-top ds′ ?Y
interpret ManiPref′: ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh ?Pd-shuffle-to-top Ch
 using TruePref.Ch-unilateral-substitutes TruePref.Ch-irc TruePref.Ch-lad TruePref.Ch-range TruePref.Ch-singular
       TruePref.Pd-shuffle-to-top-linear ManiPref.stable-on-allocation[OF ManiPref.Theorem-1[of ds]]
       TruePref.Pd-shuffle-to-top-range ManiPref.dX-range
  by unfold-locales simp-all
let ?Y′ = ManiPref′.cop ds
have ManiPref′.stable-on ds ?Y
proof(rule ManiPref′.stable-onI)
  show ManiPref′.individually-rational-on ds ?Y
  proof(rule ManiPref′.individually-rational-onI)
    show ManiPref′.CD-on ds ?Y = ?Y (is ?lhs = ?rhs)
    proof(rule set-elem-equalityI)
      fix x assume x ∈ ?rhs
      then have Xd x ∈ ds ∧ (Xd x ∉ ds′ ⟶ x ∈ Field (Pd (Xd x)))
        by (metis ManiPref.fp-cop-F-range-inv′ TruePref.CH-range′ override-on-apply-notin)
      with ManiPref.Theorem-1[of ds] ‹x ∈ ?rhs› show x ∈ ?lhs
        by (fastforce dest: ManiPref.stable-on-allocation
                   simp: ManiPref′.Cd-single ManiPref′.mem-CD-on-Cd TruePref.Pd-shuffle-to-top-Field dX-def)
    qed (meson IntE ManiPref′.CD-on-range′)
    show ManiPref′.CH ?Y = ?Y by (simp add: ManiPref′.CH-irc-idem)
  qed
  show ManiPref′.stable-no-blocking-on ds ?Y
  proof(rule ManiPref′.stable-no-blocking-onI2)
    fix h X″ assume ManiPref′.blocking-on ds ?Y h X″
    have ManiPref.blocking-on ds ?Y h X″
    proof(rule ManiPref.blocking-onI)
      fix x assume x ∈ X″
      note Pbos = ManiPref′.blocking-on-Field[OF ‹ManiPref′.blocking-on ds ?Y h X″›, where d=Xd x]
                  ManiPref′.blocking-on-allocation[OF ‹ManiPref′.blocking-on ds ?Y h X″›]
                  ManiPref′.blocking-on-CD-on′[OF ‹ManiPref′.blocking-on ds ?Y h X″› ‹x ∈ X″›]
                  ManiPref′.blocking-on-Cd[OF ‹ManiPref′.blocking-on ds ?Y h X″›, where d=Xd x]
      show x ∈ ManiPref.CD-on ds (?Y ∪ X″)
      proof(cases Xd x ∈ ds′)
        case True
        from ManiPref.fp-cop-F-allocation[of ds] ‹x ∈ X″› ‹Xd x ∈ ds′› Pbos bspec[OF ZZZ ‹Xd x ∈ ds′›]
        have dX X″ (Xd x) ⊆ Field (Pds′ (Xd x))
       by (clarsimp simp: dX-singular ManiPref′.mem-CD-on-Cd ManiPref′.Cd-Above TruePref.Pd-shuffle-to-top-Field)
          (fastforce simp: TruePref.Pd-shuffle-to-top-simps dX-singular dest: TruePref.AboveS-Pd-Xd
                     dest: ManiPref.fp-cop-F-range-inv′ ManiPref.CH-range′ TruePref.Above-shuffle-to-top)
        moreover from ManiPref.stable-on-range′[OF ManiPref.Theorem-1] ‹Xd x ∈ ds′›
        have dX ?Y (Xd x) ⊆ Field (Pds′ (Xd x))
          by (metis dX-range′ override-on-apply-in subsetI)
        moreover note bspec[OF ZZZ ‹Xd x ∈ ds′›] ‹x ∈ X″› ‹Xd x ∈ ds′› Pbos
        ultimately show ?thesis
          using ManiPref.Pd-linear′[of Xd x] ManiPref.fp-cop-F-allocation[of ds]
                ManiPref′.fp-cop-F-allocation[of ds]
          by (clarsimp simp: ManiPref′.mem-CD-on-Cd ManiPref′.Cd-Above-dX ManiPref.mem-CD-on-Cd
                            ManiPref.Cd-Above-dX dX-union dX-singular
                            TruePref.Pd-shuffle-to-top-Field TruePref.AboveS-Pd-Xd)
             (force simp: TruePref.Pd-shuffle-to-top-simps insert-absorb elim: Above-Linear-singleton
```

```
                  dest!: TruePref.Above-shuffle-to-top)
      next
        case False
        with Pbos show ?thesis
          by (fastforce simp: ManiPref'.mem-CD-on-Cd ManiPref'.Cd-greatest ManiPref.mem-CD-on-Cd
                         ManiPref.Cd-greatest TruePref.Pd-shuffle-to-top-simps)
      qed
    qed (use ‹ManiPref'.blocking-on ds ?Y h X''› in ‹simp-all add: ManiPref'.blocking-on-def›)
    with ManiPref.Theorem-1[of ds] show False by (simp add: ManiPref.blocking-on-imp-not-stable)
  qed
qed
with ManiPref'.stable-on-allocation have {x ∈ ?Y. Xd x ∈ ds'} ⊆ {x ∈ ?Y'. Xd x ∈ ds'}
  by (force dest: ManiPref'.Theorem-5[of ds]
          simp: TruePref.Pd-shuffle-to-top-simps TruePref.shuffle-to-top-def dX-def dest: inj-onD)
moreover
from ManiPref.stable-on-allocation[OF ManiPref.Theorem-1] ZZZ
have ?Z = ?Y' by (rule TruePref.Lemma-1)
moreover note YYY ZZZ
ultimately show False
  unfolding AboveS-def dX-def by (fastforce simp: ex-in-conv[symmetric] dest: TruePref.Pd-range')
qed

end
```

Again, this result does not directly apply to *gfp-F* due to the mechanism domain hypothesis.

Finally, Hatfield and Kojima (2010, Corollary 2) (respectively, Hatfield and Kojima (2009, Corollary 1)) assert that the COP (*gfp-F*) is "weakly Pareto optimal", i.e., that there is no *individually-rational* allocation that every doctor strictly prefers to the doctor-optimal match.

```
context ContractsWithUnilateralSubstitutesAndIRCAndLAD
begin

theorem Corollary-2:
  assumes ds ≠ {}
  shows ¬(∃ Y. individually-rational-on ds Y
       ∧ (∀ d∈ds. ∃ y∈Y. y ∈ AboveS (Pd d) (dX (cop ds) d)))
proof(unfold individually-rational-on-def, safe)
  fix Y assume CD-on ds Y = Y CH Y = Y
          and Z: ∀ d∈ds. ∃ y∈Y. y ∈ AboveS (Pd d) (dX (cop ds) d)
  from ‹CD-on ds Y = Y› have allocation Y by (metis CD-on-inj-on-Xd)
  from ‹CD-on ds Y = Y›
  interpret Y: ContractsWithUnilateralSubstitutesAndIRCAndLAD Xd Xh Pd-singletons-for-ds Y ds Ch
    using Ch-unilateral-substitutes Ch-irc Ch-lad Ch-range Ch-singular Pd-singletons-for-ds-range
        Pd-singletons-for-ds-linear[OF CD-on-inj-on-Xd]
    by unfold-locales (simp-all, metis)
  from Y.fp-cop-F-doctor-optimal-match Y.doctor-optimal-matchI
  have CH (Y.fp-cop-F ds) = Y
  proof(rule Y.doctor-optimal-match-unique)
    show Y.stable-on ds Y
    proof(rule Y.stable-onI)
      show Y.individually-rational-on ds Y
      proof(rule Y.individually-rational-onI)
        from ‹CD-on ds Y = Y› CD-on-Xd[where A=Y and ds=ds] show Y.CD-on ds Y = Y
          unfolding Y.CD-on-def CD-on-def
          by (force simp: Y.Cd-greatest Cd-greatest greatest-def Pd-singletons-for-ds-simps dX-def)
        from ‹CH Y = Y› show Y.CH Y = Y .
      qed
      show Y.stable-no-blocking-on ds Y
        by (rule Y.stable-no-blocking-onI,
```

```
      drule subset-trans[OF - Y.CD-on-range],
      clarsimp simp: Pd-singletons-for-ds-def dX-def Un-absorb1 subset-eq sup-commute)
  qed
 next
  fix x X assume x ∈ X Y.stable-on ds X
  with Y.Theorem-5[of ds X x] Pd-singletons-for-ds-linear[OF ‹allocation Y›]
  show ∃ y∈Y. (x, y) ∈ Pd-singletons-for-ds Y ds (Xd x)
   by (fastforce simp: Pd-singletons-for-ds-simps Y.stable-on-Xd dX-def)
 qed
 from Z ‹CH (Y.fp-cop-F ds) = Y› show False
  using group-strategy-proofD[OF
   fp-cop-F-group-strategy-proof
   ContractsWithUnilateralSubstitutesAndIRCAndLAD-axioms subset-refl
   ‹ds ≠ {}›
   Y.ContractsWithUnilateralSubstitutesAndIRCAndLAD-axioms[unfolded Pd-singletons-for-ds-def]]
  unfolding Pd-singletons-for-ds-def by force
qed

end
```

Roth and Sotomayor (1990, §4.4) discuss how the non-proposing agents can strategise to improve their outcomes in one-to-one matches.

9 Concluding remarks

We conclude with a brief and inexhaustive survey of related work.

9.1 Related work

Computer-assisted and formal reasoning. Bijlsma (1991) gives a formal pencil-and-paper derivation of the Gale-Shapley deferred-acceptance algorithm under total strict preferences and one-to-one matching (colloquially, a marriage market). He provides termination and complexity arguments, and discusses representation issues. Hamid and Castleberry (2010) treat the same algorithm in the Coq proof assistant, give a termination proof and show that it always yields a stable match. Both focus more on reasoning about programs than the theory of stable matches. Intriguingly, the latter claims that Akamai uses (modified) stable matching to assign clients to servers in their content distribution network.

Brandt and Geist (2014) use SAT technology to find results in social choice theory. They claim that the encodings used by general purpose tools like `nitpick` are too inefficient for their application.

Stable matching. In addition to the monographs Gusfield and Irving (1989); Manlove (2013); Roth and Sotomayor (1990), Roth (2008) provides a good overview up to 2007 of open problems and other aspects of this topic that we did not explore here. Sönmez and Switzer (2013) incorporate quotas and put the COP to work at the United States Military Academy. Andersson and Ehlers (2016) analyze the possibility of matching of refugees with landlords in Sweden (without mentioning matching with contracts).

One of the more famous applications of matching theory is to kidney donation (Roth 2015), a *repugnant market* where the economists' basic tool of pricing things is considered verboten. These markets are sometimes, but not always, two-sided – kidneys are often exchanged due to compatibility issues, but there are also altruistic donations and recipients who cannot reciprocate – and so the model we discussed here is not applicable. Instead generalizations of Gale's *top trading cycles* algorithm are pressed into service (Abdulkadirolu and Sönmez 1999; Shapley and Scarf 1974; Sönmez and Ünver 2010). Much recent work has hybridized these approaches – for instance, Dworczak (2016) uses a combination to enumerate all stable matches.

Echenique (2012) shows that the matching with contracts model of §5 is no more general than that of Kelso and Crawford (1982) (a job matching market with salaries). Schlegel generalizes this result to the COP setting of §6, and moreover shows how lattice structure can be recovered there, which yields a hospital-proposing deferred-acceptance algorithm that relies only on unilaterally substitutable hospital choice functions. See Hatfield and Kominers (2016) for a discussion of the many-to-many case.

Roth and Sotomayor (1990, Theorem 2.33) point to alternatives to the deferred-acceptance algorithm, and to more general matching scenarios involving couples and roommates. Manlove (2013) provides a comprehensive survey of matching with preferences.

Further results: COP. Afacan (2014) explores the following two properties:

> *[Population monotonicity]* says that no doctor is to be worse off whenever some others leave the market. *[Resource monotonicity]*, on the other hand, requires that no doctor should lose whenever hospitals start hiring more doctors.

He shows that the COP is population and resource monotonic under *irc* and *bilateral_substitutes*. Also Afacan (2015) characterizes the COP by the properties *truncation proof* ("no doctor can ever benefit from truncating his preferences") and *invariant to lower tail preferences change* ("any doctor's assignment does not depend on his preferences over worse contracts"); that the COP satisfies these properties was demonstrated in §6. See also Hatfield et al. (2016) for another set of conditions that characterize the COP.

Hirata and Kasuya (2016) show how the strategic results can be obtained without the rural hospitals theorem, in a setting that requires *irc* but not substitutability.

Further results: Strategy. There are many different ways to think about the manipulation of economic mechanisms. Some continue in the game-theoretic tradition (Gonczarowski 2014), and, for instance, compare the manipulability of mechanisms that yield stable matches (Chen et al. 2016). Techniques from computer science help refine the notion of strategy-proofness (Ashlagi and Gonczarowski 2015) and enable complexity-theoretic arguments (Aziz et al. 2015; Deng et al. 2016). Kojima and Pathak (2009) have analyzed the scope for manipulation in large matching markets.

10 Acknowledgements

I thank Rob Irving for a copy of his excellent monograph (Gusfield and Irving 1989), Jasmin C. Blanchette for help with nitpick, Andreas Lochbihler for his Bourbaki-Witt Fixpoint theory, Orhan Aygün for a helpful discussion of Aygün and Sönmez (2012a), and Roman Werpachowski for general comments.

References

A. Abdulkadirolu and T. Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233 – 260, 1999. doi: 10.1006/jeth.1999.2553.

M. O. Afacan. Some further properties of the cumulative offer process. Unpublished, August 2014. URL http://ssrn.com/abstract=2478418.

M. O. Afacan. Characterizations of the cumulative offer process. Unpublished, August 2015. URL http://ssrn.com/abstract=2653751.

M. O. Afacan and B. Turhan. On relationships between substitutes conditions. *Economics Letters*, 126:10 – 12, 2015. doi: 10.1016/j.econlet.2014.11.007.

M. Aizerman and A. Malishevski. General theory of best variants choice: Some aspects. *IEEE Transactions on Automatic Control*, 26(5):1030–1040, October 1981. doi: 10.1109/TAC.1981.1102777.

T. Andersson and L. Ehlers. Assigning refugees to landlords in Sweden: Stable maximum matchings. Working Paper 2016:18, Department of Economics, School of Economics and Management, Lund University, July 2016. URL http://project.nek.lu.se/publications/workpap/papers/wp16_18.pdf.

I. Ashlagi and Y. A. Gonczarowski. No stable matching mechanism is obviously strategy-proof. *CoRR*, abs/1511.00452, 2015. URL http://arxiv.org/abs/1511.00452.

O. Aygün and T. Sönmez. The importance of irrelevance of rejected contracts in matching under weakened substitutes conditions. Working Paper, June 2012a.

O. Aygün and T. Sönmez. Matching with contracts: The critical role of irrelevance of rejected contracts. Working Paper. Abridged in AER 103(5) as "Matching with Contracts: Comment" in 2013., May 2012b.

H. Aziz, H. G. Seedig, and J. K. von Wedel. On the susceptibility of the deferred acceptance algorithm. In G. Weiss, P. Yolum, R. H. Bordini, and E. Elkind, editors, *AAMAS 2015*, pages 939–947. ACM, 2015.

L. Bijlsma. Formal derivation of a stable marriage algorithm. In W.H.J. Feijen and A.J.M van Gastreren, editors, *C.S. Scholten dedicata: van oude machines en nieuwe rekenwijzen*. Academic Service Schoonhoven, 1991.

J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 9 (1):101–148, 2016. doi: 10.6092/issn.1972-5787/4593.

K. C. Border. Introductory notes on preference and rational choice, April 2012. URL http://people.hss.caltech.edu/~kcb/Notes.shtml.

F. Brandt and C. Geist. Finding strategyproof social choice functions via SAT solving. In A. L. C. Bazzan, M. N. Huhns, A. Lomuscio, and P. Scerri, editors, *AAMAS '14*, pages 1193–1200. IFAAMAS/ACM, 2014.

L. Bulwahn, A. Krauss, F. Haftmann, L. Erkök, and J. Matthews. Imperative functional programming with Isabelle/HOL. In O. A. Mohamed, C. Muñoz, and S. Tahar, editors, *TPHOLs '08*, volume 5170 of *LNCS*, pages 352–367. Springer, 2008.

M. B. Caminati, M. Kerber, C. Lange, and C. Rowat. VCG — combinatorial Vickrey-Clarke-Groves auctions. *Archive of Formal Proofs*, 2015, 2015a. URL https://isa-afp.org/entries/Vickrey_Clarke_Groves.shtml.

M. B. Caminati, M. Kerber, C. Lange, and C. Rowat. Sound auction specification and implementation. In T. Roughgarden, M. Feldman, and M. Schwarz, editors, *EC '15*, pages 547–564, New York, NY, USA, 2015b. ACM. doi: 10.1145/2764468.2764511.

C. P. Chambers and M. B. Yenmez. Choice and matching. Under review, July 2013. URL http://econweb.ucsd.edu/~cpchambers/papers.html.

P. Chen, M. Egesdal, M. Pycia, and M. B. Yenmez. Manipulability of stable mechanisms. *American Economic Journal: Microeconomics*, 8(2):202–14, May 2016. doi: 10.1257/mic.20150035.

A. Ciupan, J. W. Hatfield, and S. D. Kominers. An elementary proof of the lone wolf theorem. Unpublished, August 2016.

Y. Deng, W. Shen, and P. Tang. Coalition manipulations of the gale-shapley algorithm. *CoRR*, abs/1502.07823, 2016. URL http://arxiv.org/abs/1502.07823.

P. Dworczak. Deferred acceptance with compensation chains. In V. Conitzer, D. Bergemann, and Y. Chen, editors, *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 65–66. ACM, 2016. doi: 10.1145/2940716.2940727.

F. Echenique. Contracts versus salaries in matching. *American Economic Review*, 102(1):594–601, 2012. doi: 10.1257/aer.102.1.594.

T. Fleiner. *Stable and Crossing Structures*. PhD thesis, Technische Universiteit Eindhoven, 2000.

T. Fleiner. Some results on stable matchings and fixed points. Technical Report tr-2002-08, Egerváry Research Group on Combinatorial Optimization, Budapest, Hungary, March 2002. URL http://www.cs.elte.hu/egres/www/tr-02-08.html.

T. Fleiner. A fixed-point approach to stable matchings and some applications. *Mathematics of Operations Research*, 28(1):103–126, 2003. doi: 10.1287/moor.28.1.103.14256. URL http://www.cs.elte.hu/egres/www/tr-01-01.html.

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, January 1962.

Y. A. Gonczarowski. Manipulation of stable matchings using minimal blacklists. In M. Babaioff, V. Conitzer, and D. Easley, editors, *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA, June 8-12, 2014*, page 449. ACM, 2014. doi: 10.1145/2600057.2602840.

D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989.

F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In M. Blume, N. Kobayashi, and G. Vidal, editors, *FLOPS*, volume 6009 of *LNCS*. Springer, 2010.

N. A. Hamid and C. Castleberry. Formally certified stable marriages. In H. C. Cunningham, P. Ruth, and N. A. Kraft, editors, *Proceedings of the 48th Annual Southeast Regional Conference, 2010, Oxford, MS, USA, April 15-17, 2010*, page 34. ACM, 2010. doi: 10.1145/1900008.1900056.

S. O. Hansson and T. Grüne-Yanoff. Preferences. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford, winter 2012 edition, 2012. URL http://plato.stanford.edu/archives/win2012/entries/preferences/.

J. W. Hatfield and F. Kojima. Matching with contracts: Comment. *American Economic Review*, 98(3):1189–94, 2008. doi: 10.1257/aer.98.3.1189.

J. W. Hatfield and F. Kojima. Group incentive compatibility for matching with contracts. *Games and Economic Behavior*, 67(2):745–749, 2009. doi: 10.1016/j.geb.2009.01.007.

J. W. Hatfield and F. Kojima. Substitutes and stability for matching with contracts. *Journal of Economic Theory*, 145(5):1704–1723, 2010. doi: 10.1016/j.jet.2010.01.007.

J. W. Hatfield and P. R. Milgrom. Matching with contracts. *American Economic Review*, 95(4):913–935, 2005. doi: 10.1257/0002828054825466.

J. W. Hatfield, S. D. Kominers, and A. Westkamp. Stability, strategy-proofness, and cumulative offer mechanisms. presented at MATCHUP 2015, July 2016.

J.W. Hatfield and S.D. Kominers. Contract design and stability in many-to-many matching. *Games and Economic Behavior*, 2016. doi: 10.1016/j.geb.2016.01.002.

D. Hirata and Y. Kasuya. Cumulative offer process is order-independent. *Economics Letters*, 124(1):37 – 40, 2014. doi: 10.1016/j.econlet.2014.04.008.

D. Hirata and Y. Kasuya. On stable and strategy-proof rules in matching markets with contracts. Unpublished, August 2016.

A. S. Kelso and V. P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50(6): 1483–1504, November 1982.

F. Kojima. Impossibility of stable and nonbossy matching mechanisms. *Economics Letters*, 107(1):69 – 70, 2010. doi: 10.1016/j.econlet.2009.12.029.

F. Kojima and P. A. Pathak. Incentives and stability in large two-sided matching markets. *American Economic Review*, 99(3):608–627, 2009. doi: 10.1257/aer.99.3.608.

D. F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science*. WorldScientific, 2013. doi: 10.1142/8591.

H. Moulin. Choice functions over a finite set: a summary. *Social Choice and Welfare*, 2(2):147–160, September 1985. doi: 10.1007/BF00437315. URL http://eprints.gla.ac.uk/92191/.

T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, Berlin, 2002.

C. R. Plott. Path independence, rationality and social choice. *Econometrica*, 41(6):1075–1091, 1973.

A. E. Roth. *Who gets what – and why: the new economics of matchmaking and market design*. Houghton Mifflin Harcourt, 2015.

A. E. Roth and M. A. Sotomayor. *Two-sided matching: A Study in Game-Theoretic Modeling and Analysis*. Econometric society monographs. Cambridge University Press, Cambridge, United Kingdom, 1990.

Alvin Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *International Journal of Game Theory*, 36(3):537–569, March 2008. URL http://www.nber.org/papers/w13225.

J. C. Schlegel. Contracts versus salaries in matching: A general result. *Journal of Economic Theory*, 159, Part A: 552–573, 2015. doi: 10.1016/j.jet.2015.07.015.

A. K. Sen. *Collective Choice and Social Welfare*. Holden Day, 1970.

A. K. Sen. Choice functions and revealed preference. *The Review of Economic Studies*, 38(3):307–317, July 1971.

L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974. doi: 10.1016/0304-4068(74)90033-0.

M. Sotomayor. A non-constructive elementary proof of the existence of stable marriages. *Games and Economic Behavior*, 13(1):135 – 137, 1996. doi: 10.1006/game.1996.0029.

T. Sönmez and T. B. Switzer. Matching with (branch-of-choice) contracts at the United States Military Academy. *Econometrica*, 81(2):451–488, 2013. doi: 10.3982/ECTA10570.

T. Sönmez and M. U. Ünver. House allocation with existing tenants: A characterization. *Games and Economic Behavior*, 69(2):425 – 445, 2010. doi: 10.1016/j.geb.2009.10.010.