Implementing the Omega Test in HOL

Outline:

- Basic Fourier-Motzkin variable elimination
- Omega’s extension to F-M variable elimination
- Implementing this in HOL
- On the need for efficiency in conversion to DNF
Fourier-Motzkin Variable Elimination

The basis for Hodes’s method (ARITH_CONV in HOL and d.p.’s in Isabelle, ACL2 and Coq)

Fundamental theorem:

\[(\exists x. a \leq \alpha x \land \beta x \leq b) \equiv a\beta \leq \alpha b\]

True over \(\mathbb{R}\) (and \(\mathbb{Q}\))…
Fourier-Motzkin Variable Elimination

- The basis for Hodes’s method (ARITH_CONV in HOL and d.p.’s in Isabelle, ACL2 and Coq)
- Fundamental theorem:

\[(\exists x. \ a \leq \alpha x \land \beta x \leq b) \equiv a\beta \leq \alpha b\]

- True over \(\mathbb{R}\) (and \(\mathbb{Q}\))...
- ...false over \(\mathbb{Z}\)

  E.g., \((\exists x. \ 3 \leq 2x \leq 3) \not\equiv 6 \leq 6\)
Let $L(x)$ be conjunction of lower bounds on $x$, indexed by $i$, of the form $a_i \leq \alpha_i x \ (\alpha_i > 0)$. 

Let $U(x)$ be conjunction of upper bounds on $x$, indexed by $j$, of the form $\beta_j x \leq b_j \ (\beta_j > 0)$. 

Want to show:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

On assumption that core theorem is true. (Similar “extension to $n \times m$ constraints” proofs are required for theorems over $\mathbb{Z}$.)
Multiple Constraints: Induction #1

Many upper bounds, one lower bound. Have:

\[
(\exists x. \ a \leq \alpha x \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j
\]

Want

\[
(\exists x. \ a \leq \alpha x \land \beta x \leq b \land U(x)) \equiv \\
\bigwedge_j a\beta_j \leq \alpha b_j \land a\beta \leq \alpha b
\]
Multiple Constraints: Induction #1

Many upper bounds, one lower bound. Have:

\[(\exists x. \ a \leq \alpha x \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j\]

Want

\[(\exists x. \ a \leq \alpha x \land \beta x \leq b \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j \land a\beta \leq \alpha b\]

Left to right is easy: I.H. gives first conjunct; core theorem gives second.
**Multiple Constraints: Induction #1**

Many upper bounds, one lower bound. Have:

\[
(\exists x. \ a \leq \alpha x \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j
\]

Want

\[
(\exists x. \ a \leq \alpha x \land \beta x \leq b \land U(x)) \equiv \\
\bigwedge_j a\beta_j \leq \alpha b_j \land a\beta \leq \alpha b
\]

Right to left: I.H. gives us \(\exists y. \ a \leq \alpha y \land U(y)\)
Many upper bounds, one lower bound. Have:

\[(\exists x. \ a \leq \alpha x \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j\]

Want

\[(\exists x. \ a \leq \alpha x \land \beta x \leq b \land U(x)) \equiv \bigwedge_j a\beta_j \leq \alpha b_j \land a\beta \leq \alpha b\]

Right to left: I.H. gives us \(\exists y. \ a \leq \alpha y \land U(y)\)
Core theorem gives \(\exists z. \ a \leq \alpha z \land \beta z \leq b\)
Many upper bounds, one lower bound. Have:

\[(\exists x. \ a \leq \alpha x \land U(x)) \equiv \bigwedge_{j} a\beta_{j} \leq \alpha b_{j}\]

Want

\[(\exists x. \ a \leq \alpha x \land \beta x \leq b \land U(x)) \equiv \bigwedge_{j} a\beta_{j} \leq \alpha b_{j} \land a\beta \leq \alpha b\]

Right to left: I.H. gives us \(\exists y. \ a \leq \alpha y \land U(y)\)
Core theorem gives \(\exists z. \ a \leq \alpha z \land \beta z \leq b\)
y and z both satisfy \((a, \alpha)-constraint\). Minimum of y and z will satisfy both upper-bound constraints.
Multiple Constraints: Induction #2

$n$ upper bounds, $m$ lower bounds. Have:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

Want

$$(\exists x. \ a \leq \alpha x \land L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \land \bigwedge_j a \beta_j \leq \alpha b_j$$
Multiple Constraints: Induction #2

$n$ upper bounds, $m$ lower bounds. Have:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

Want

$$(\exists x. \ a \leq \alpha x \land L(x) \land U(x)) \equiv \\
\bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \land \bigwedge_j a \beta_j \leq \alpha b_j$$

Left to right: first conjunct by I.H.; second by appeal to induction #1
Multiple Constraints: Induction #2

$n$ upper bounds, $m$ lower bounds. Have:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

Want

$$(\exists x. \ a \leq \alpha x \land L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \land \bigwedge_j a \beta_j \leq \alpha b_j$$

Right to left: I.H. gives $\exists y. \ L(y) \land U(y)$. 

ARG lunch – p.7
Multiple Constraints: Induction #2

$n$ upper bounds, $m$ lower bounds. Have:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

Want

$$\left( \exists x. \ a \leq \alpha x \land L(x) \land U(x) \right) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \land \bigwedge_j a \beta_j \leq \alpha b_j$$

Right to left: I.H. gives $\exists y. \ L(y) \land U(y)$. Induction #1 gives $\exists z. \ a \leq \alpha z \land U(z)$.  

ARG lunch – p.7
Multiple Constraints: Induction #2

$n$ upper bounds, $m$ lower bounds. Have:

$$(\exists x. \ L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i \beta_j$$

Want

$$(\exists x. \ a \leq \alpha x \land L(x) \land U(x)) \equiv \bigwedge_{i,j} a_i \beta_j \leq \alpha_i \beta_j \land \bigwedge_j a \beta_j \leq \alpha \beta_j$$

Right to left: I.H. gives $\exists y. \ L(y) \land U(y)$.
Induction #1 gives $\exists z. \ a \leq \alpha z \land U(z)$.

$y$ and $z$ both satisfy $U$. Take their maximum to satisfy $L$ and the other lower bound constraint.
Exact Shadow Elimination

The formula

\[ \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \]

is known as the real shadow. If all of the \( \alpha_i \) or all of the \( \beta_j \) are equal to 1, then we can use it to eliminate quantifiers over \( \mathbb{Z} \).
Exact Shadow Elimination

The formula

$$\bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

is known as the real shadow. If all of the $\alpha_i$ or all of the $\beta_j$ are equal to 1, then we can use it to eliminate quantifiers over $\mathbb{Z}$.

The core theorem

$$(\exists x. a \leq \alpha x \land \beta x \leq b) \equiv a \beta \leq \alpha b$$

is true over $\mathbb{Z}$ because...
Exact Shadow Elimination

The formula

$$\bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j$$

is known as the *real shadow*. If all of the $\alpha_i$ or all of the $\beta_j$ are equal to 1, then we can use it to eliminate quantifiers over $\mathbb{Z}$.

The core theorem

$$(\exists x. \ a \leq \alpha x \land \beta x \leq b) \equiv a\beta \leq \alpha b$$

is true over $\mathbb{Z}$ because...

- left to right: transitivity still holds
The formula
\[ \bigwedge_{i,j} a_i \beta_j \leq \alpha_i b_j \]
is known as the real shadow. If all of the \( \alpha_i \) or all of the \( \beta_j \) are equal to 1, then we can use it to eliminate quantifiers over \( \mathbb{Z} \).

The core theorem
\[ (\exists x. a \leq \alpha x \land \beta x \leq b) \equiv a \beta \leq \alpha b \]
is true over \( \mathbb{Z} \) because...

- left to right: transitivity still holds
- right to left: take \( x = b \) if \( \beta = 1 \), \( x = a \) if \( \alpha = 1 \)
Pugh claims that exact shadow eliminations occur frequently. Otherwise, following theorem required:

Let $m$ be the maximum of all the $\beta_j$ s. Then

\[
(\exists x. \ L(x) \land U(x)) \equiv \\
(\land_{i,j}(\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \\
\land \\
\lor \\
\lor_{i=0}^{m} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)
\]

First disjunct known as *dark shadow*. Other disjuncts known as *splinters*.
Proof of Core Omega Theorem

Result is of form

\[(\exists x. L(x) \land U(x)) \equiv \text{“dark shadow”} \lor \text{“splinters”}\]

Proof has three cases:

- “dark shadow” \(\Rightarrow\) \(\exists x. L(x) \land U(x)\)
- “splinters” \(\Rightarrow\) \(\exists x. L(x) \land U(x)\)
- \((\exists x. L(x) \land U(x)) \land \neg \text{“dark shadow”} \Rightarrow \text{“splinters”}\)
Core Omega Theorem—Case 1

\[ \land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \land U(x) \]
Core Omega Theorem—Case 1

\[ \land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \Rightarrow \exists x. L(x) \land U(x) \]

Do singleton case, extend by two inductions as before:

\[ (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \Rightarrow \exists x. a \leq \alpha x \land \beta x \leq b \]
Core Omega Theorem—Case 1

\[ \land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \land U(x) \]

Do singleton case, extend by two inductions as before:

\[ (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \implies \exists x. a \leq \alpha x \land \beta x \leq b \]

Assume opposite, so \( \neg \exists x. \ a \beta \leq \alpha \beta x \leq \alpha b \)
Core Omega Theorem—Case 1

\( \wedge_{i,j}(\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \Rightarrow \exists x. L(x) \wedge U(x) \)

Do singleton case, extend by two inductions as before:

\( (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \Rightarrow \exists x. a \leq \alpha x \wedge \beta x \leq b \)

Assume opposite, so \( \neg \exists x. a \beta \leq \alpha \beta x \leq \alpha b \)

No multiple of \( \alpha \beta \) between \( a \beta \) and \( \alpha b \), so

\( \exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1) \)
Core Omega Theorem—Case 1

\( \wedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \wedge U(x) \)

Do singleton case, extend by two inductions as before:

\((\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \implies \exists x. a \leq \alpha x \wedge \beta x \leq b\)

Assume opposite, so \(\neg \exists x. a \beta \leq \alpha \beta x \leq \alpha b\)

No multiple of \(\alpha \beta\) between \(a \beta\) and \(\alpha b\), so

\(\exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1)\)

Have \(0 < \alpha \beta (i + 1) - \alpha b\)
Core Omega Theorem—Case 1

\[ \land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \land U(x) \]

Do singleton case, extend by two inductions as before:

\( (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \implies \exists x. a \leq \alpha x \land \beta x \leq b \)

Assume opposite, so \( \neg \exists x. a \beta \leq \alpha \beta x \leq \alpha b \)

No multiple of \( \alpha \beta \) between \( a \beta \) and \( \alpha b \), so

\( \exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1) \)

Have \( 0 < \alpha \beta (i + 1) - \alpha b \), so \( 1 \leq \beta (i + 1) - b \)
Core Omega Theorem—Case 1

\( \land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \Rightarrow \exists x. L(x) \land U(x) \)

Do singleton case, extend by two inductions as before:

\( (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \Rightarrow \exists x. a \leq \alpha x \land \beta x \leq b \)

Assume opposite, so \( \neg \exists x. \alpha \beta \leq \alpha \beta x \leq \alpha b \)

No multiple of \( \alpha \beta \) between \( a \beta \) and \( \alpha b \), so

\( \exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1) \)

Have \( 0 < \alpha \beta (i + 1) - \alpha b \), so \( 1 \leq \beta (i + 1) - b \), so

\( \alpha \leq \alpha \beta (i + 1) - \alpha b. \)
Core Omega Theorem—Case 1

\(\bigwedge_{i,j}(\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \land U(x)\)

Do singleton case, extend by two inductions as before:

\((\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \implies \exists x. a \leq \alpha x \land \beta x \leq b\)

Assume opposite, so \(\neg \exists x. a \beta \leq \alpha \beta x \leq \alpha b\)

No multiple of \(\alpha \beta\) between \(a \beta\) and \(\alpha b\), so

\(\exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta(i + 1)\)

Have \(0 < \alpha \beta(i + 1) - \alpha b\), so \(1 \leq \beta(i + 1) - b\), so \(\alpha \leq \alpha \beta(i + 1) - \alpha b\). Similarly, \(\beta \leq a \beta - \alpha \beta i\).
Core Omega Theorem—Case 1

\[(\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \implies \exists x. L(x) \land U(x)\]

Do singleton case, extend by two inductions as before:

\[(\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \implies \exists x. a \leq \alpha x \land \beta x \leq b\]

Assume opposite, so \(\neg \exists x. a \beta \leq \alpha \beta x \leq \alpha b\)

No multiple of \(\alpha \beta\) between \(a \beta\) and \(\alpha b\), so

\(\exists i. \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1)\)

Have \(0 < \alpha \beta (i + 1) - \alpha b\), so \(1 \leq \beta (i + 1) - b\), so

\(\alpha \leq \alpha \beta (i + 1) - \alpha b\). Similarly, \(\beta \leq a \beta - \alpha \beta i\).

Thus, \(\alpha + \beta \leq \alpha \beta + a \beta - \alpha b\).
Core Omega Theorem—Case 1

\[ \forall i,j (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j \Rightarrow \exists x. \ L(x) \land U(x) \]

Do singleton case, extend by two inductions as before:

\[ (\alpha - 1)(\beta - 1) \leq \alpha b - a \beta \Rightarrow \exists x. \ a \leq \alpha x \land \beta x \leq b \]

Assume opposite, so \( \neg \exists x. \ a \beta \leq \alpha \beta x \leq \alpha b \)

No multiple of \( \alpha \beta \) between \( a \beta \) and \( \alpha b \), so

\[ \exists i. \ \alpha \beta i < a \beta \leq \alpha b < \alpha \beta (i + 1) \]

Have \( 0 < \alpha \beta (i + 1) - \alpha b \), so \( 1 \leq \beta (i + 1) - b \), so

\( \alpha \leq \alpha \beta (i + 1) - \alpha b \). Similarly, \( \beta \leq a \beta - \alpha \beta i \).

Thus, \( \alpha + \beta \leq \alpha \beta + a \beta - \alpha b \).

Rearrange to \( \alpha b - a \beta < \alpha \beta - \alpha - \beta + 1 \).

Contradicts assumption!
Core Omega Theorem—Case 2

\[ \bigvee_i \bigvee_{k=0}^{\left\lfloor \frac{m\alpha_i - \alpha_i - m}{m} \right\rfloor} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x) \]

\[ \Rightarrow \]

\[ \exists x. L(x) \land U(x) \]
Core Omega Theorem—Case 2

\[ \forall_i \left( \bigvee_{k=0}^{\left\lfloor \frac{m\alpha_i - \alpha_i - m}{m} \right\rfloor} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x) \right) \Rightarrow \exists x. L(x) \land U(x) \]

Trivial, each disjunct provides a witness that satisfies \( L \) and \( U \).
Core Omega Theorem—Case 3

\[(\exists x. L(x) \land U(x)) \land \neg(\land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \forall i \forall_{k=0}^{m} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)\]
Core Omega Theorem—Case 3

\((\exists x. L(x) \land U(x)) \land \neg(\bigwedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \bigvee_i \bigvee_{k=0}^{m \alpha_i - \alpha_i - m} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)\)

Negated second assumption means
\[ \alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U) \]
Core Omega Theorem—Case 3

\[(\exists x. L(x) \land U(x)) \land \neg(\wedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \]

\[\forall i \forall \sum_{k=0}^{m} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)\]

Negated second assumption means

\[\alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U)\]

From assumption #1, \(\beta x \leq b\)
Core Omega Theorem—Case 3

$$(\exists x. \, L(x) \land U(x)) \land \neg(\land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_ib_j - a_i\beta_j) \Rightarrow$$

$$\forall i \forall k=0 \left\lfloor \frac{m\alpha_i - \alpha_i - m}{m} \right\rfloor \exists x. \, (\alpha_i x = a_i + k) \land L(x) \land U(x)$$

Negated second assumption means

$$\alpha b - a\beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U)$$

From assumption #1, $\beta x \leq b$, so $\alpha \beta x \leq \alpha b$. 
Core Omega Theorem—Case 3

\[(\exists x. L(x) \land U(x)) \land \neg(\land_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \]

\[\bigvee_i \bigvee_{k=0}^{\frac{m\alpha_i - \alpha_i - m}{m}} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)\]

Negated second assumption means

\[\alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U)\]

From assumption #1, \(\beta x \leq b\), so \(\alpha \beta x \leq \alpha b\).

Combining,

\[\alpha \beta x \quad \leq \quad a \beta + \alpha \beta - \beta - \alpha\]
Core Omega Theorem—Case 3

\( (\exists x. L(x) \land U(x)) \land \neg(\bigwedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \)

\[ \forall i \forall k=0^{[\frac{m\alpha_i-\alpha_i-m}{m}]} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x) \]

Negated second assumption means

\( \alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U) \)

From assumption #1, \( \beta x \leq b \), so \( \alpha \beta x \leq \alpha b \).

Combining,

\( \alpha \beta x \leq \alpha \beta + \alpha \beta - \beta - \alpha \)

\( \Rightarrow \beta(\alpha x - a) \leq \alpha \beta - \beta - \alpha \)
Core Omega Theorem—Case 3

\[(\exists x. L(x) \land U(x)) \land \neg(\bigwedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i \beta_j - a_i \beta_j) \Rightarrow \]
\[\forall i \forall k=0 \left( \frac{\alpha_i - \alpha_i - m}{m} \right) \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x) \]

Negated second assumption means
\[\alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{ and } b \text{ from } L, U)\]

From assumption #1, \(\beta x \leq b\), so \(\alpha \beta x \leq \alpha b\).
Combining,
\[\Rightarrow \beta (\alpha x - a) \leq \alpha \beta - \beta - \alpha\]
\[\Rightarrow \alpha x - a \leq \left[ \frac{\alpha \beta - \beta - \alpha}{\beta} \right] \]
Core Omega Theorem—Case 3

\[(\exists x. L(x) \land U(x)) \land \neg(\bigwedge_{i,j} (\alpha_i - 1)(\beta_j - 1) \leq \alpha_i b_j - a_i \beta_j) \Rightarrow \forall_i \forall_{k=0}^{\text{min}(\frac{m\alpha_i - \alpha_i - m}{m})} \exists x. (\alpha_i x = a_i + k) \land L(x) \land U(x)\]

Negated second assumption means
\[\alpha b - a \beta \leq \alpha \beta - \beta - \alpha \quad (\alpha, \beta, a, \text{and } b \text{ from } L, U)\]

From assumption #1, \(\beta x \leq b\), so \(\alpha \beta x \leq \alpha b\).

Combining,
\[\alpha \beta x \leq \alpha \beta + \alpha \beta - \beta - \alpha\]
\[\Rightarrow \beta (\alpha x - a) \leq \alpha \beta - \beta - \alpha\]
\[\Rightarrow \alpha x - a \leq \left[\frac{\alpha \beta - \beta - \alpha}{\beta}\right] \leq \left[\frac{m\alpha - \alpha - m}{m}\right]\]

Can now pick appropriate splinter-disjunct and we’re done.
Splinters’ Equality Constraints

Splinters’ $\exists x. (cx = e) \land \ldots$ can be eliminated:

- Multiply through other terms to include $cx$, which sub-terms are replaced by $e$
- Pull in quantifier, and formula becomes $c|e$. 

ARG lunch – p.14
To eliminate $c | (dx + e)$ with $x$ existentially quantified:

- Reduce all coefficients on RHS by taking “mod $c$”
- Introduce fresh existential variable $y$:
  $$\exists y. \, cy = dx + e$$
- $d < c$, so eliminate $x$, to get $d | e - cy$
- Iterate...
Eliminating Divisibility Constraints

To eliminate $c|(dx + e)$ with $x$ existentially quantified:

- Reduce all coefficients on RHS by taking “mod $c$”
- Introduce fresh existential variable $y$:
  \[
  \exists y. \; cy = dx + e
  \]
- $d < c$, so eliminate $x$, to get $d|e - cy$
- Iterate...

To eliminate $\neg(c|e)$ convert to

\[
\bigvee_{i=1}^{c-1} c|(e + i)
\]
Implementation #1

- Prove core theorems in HOL
- Code for decision procedure is specialised rewriting engine
  - it transforms equivalent terms to equivalent terms
- All the work is done in the logic
Implementation #1 (continued)

- Hard to prove facts or define functions over formulas directly.
  - Formulas would have type \( \text{:bool} \), or perhaps \( \text{:int} \rightarrow \text{bool} \).

- Instead, define special syntax and evaluation functions.

- I have two functions, `evallower` and `evalupper`, both of type
  \( \text{:int} \rightarrow (\text{num} \# \text{int}) \text{ list} \rightarrow \text{bool} \).

- These provide a concrete (or “shadow”) syntax to manipulate.
Evaluation Functions

Defining equations:

\[ \text{evallower} \ x \ [] = T \]
\[ \text{evallower} \ x \ ((c,y)::cs) = \]
\[ \quad y \leq &c \ast x \land \text{evallower} \ x \ cs \]

\[ \text{evalupper} \ x \ [] = T \]
\[ \text{evalupper} \ x \ ((c,y)::cs) = \]
\[ \quad &c \ast x \leq y \land \text{evalupper} \ x \ cs \]
Now some arbitrary HOL formula

\[ \forall x. 3 + y \leq x \ldots \]

can be converted into a standard form

\[ \forall x. \text{evallower } x \text{ lows } \land \text{evalupper } x \text{ ups} \]

and most importantly, it’s a theorem:

\[ \vdash \text{<oldform> } = \forall x. \text{evallower } \ldots \]

Now appeal to the core theorem...
Core Theorems in HOL

With splinters (the “nightmare scenario”):

\[- \forall \text{uppers lowers } m. \]
\[\text{EVERY } \text{fst}_n\text{zero uppers} \land \text{EVERY } \text{fst}_n\text{zero lowers} \land \]
\[\text{EVERY } (\forall p. \text{FST } p \leq m) \text{ uppers} \implies \]
\[(\exists x. \text{evalupper } x \text{ uppers} \land \text{evallower } x \text{ lowers}) = \]
\[\text{dark}_\text{shadow} \text{ uppers lowers} \lor \]
\[\exists x. \text{nightmare } x \text{ m uppers lowers lowers} \]

Or, if exact shadow elimination is possible, use

\[- \forall \text{uppers lowers}. \]
\[\text{EVERY } \text{fst}_n\text{zero uppers} \land \text{EVERY } \text{fst}_n\text{zero lowers} \implies \]
\[\text{EVERY } \text{fst1 uppers} \lor \text{EVERY } \text{fst1 lowers} \implies \]
\[(\exists x. \text{evalupper } x \text{ uppers} \land \text{evallower } x \text{ lowers}) = \]
\[\text{real}_\text{shadow} \text{ uppers lowers} \]
Calculating Shadows in the Logic

Two characterisations of real_shadow:

■ Logical:

\[ \text{|- real_shadow uppers lowers} = \]
\[ !c \, d \, L \, R. \]
\[ \text{MEM} (c,L) \, \text{uppers} \land \text{MEM} (d,R) \, \text{lowers} \implies \]
\[ &c \times R \leq &d \times L \]

■ Made for rewriting:

\[ \text{|- (real_shadow} [\] \, \text{lowers} = \text{T}) \land \]
\[ (\text{real_shadow} (\text{upper::us}) \, \text{lowers} = \]
\[ \text{rshadow_row} \, \text{upper} \, \text{lowers} \land \]
\[ \text{real_shadow} \, \text{us} \, \text{lowers}) \]
\[ \text{|- (rshadow_row} (\text{upc,upy}) [\] = \text{T}) \land \]
\[ (\text{rshadow_row} (\text{upc,upy}) ((\text{lowc,lowy})::rs) = \]
\[ &\text{upc} \times \text{lowy} \leq &\text{lowc} \times \text{upy} \land \]
\[ \text{rshadow_row} (\text{upc,upy}) \, \text{rs}) \]
Implementation #2

Design philosophy:

- For purely existential or universal goals
- If goal is unsatisfiable, find a proof of contradiction outside the logic
- If goal is satisfiable, find satisfying assignment outside the logic.

This approach can “win big” because every variable elimination turns $n$ constraints into $O(n^2)$ new constraints.

Having to manage this explosion in the logic is costly, particularly as most of the work is redundant.
Implementation #2: Details

Calculate the real shadow for all of the variables. If this is false, so too is the original (existential) goal. If inexact, then calculate dark shadow. If this provides satisfying assignment, we’re done. Otherwise, must resort to (symbolic) splinters.

Calculate new constraints outside the logic, but accompany each with a data structure that would allow it to be proved if necessary:

```plaintext
datatype derivation =
    ASM of term
  | REAL_COMBIN of int * derivation * derivation
  | GCD_CHECK of derivation
  | DIRECT_CONTR of derivation * derivation
```

(Richard’s ARITH_CONV uses closures here.)
If the system is reduced to one variable $x$, it will have at most two constraints: $x \leq c$ and $d \leq x$. If $d \leq c$ return satisfying assignment with $x \leftarrow c$. Recurse back through other variables. Otherwise, derive contradiction.
If the system is reduced to one variable $x$, it will have at most two constraints: $x \leq c$ and $d \leq x$. If $d \leq c$ return satisfying assignment with $x \mapsto c$. Recurse back through other variables. Otherwise, derive contradiction.

If the system ever has a variable $x$ with only upper (lower) bounds, return satisfying assignment with all other present variables to zero, and $x$ to minimum (maximum) of resulting constants. Recurse back through other variables.
Interpreting Results

- If doing exact shadow elimination, both satisfying assignments and contradictions are valid.
- If doing a real (but inexact) shadow, only contradictions make sense.
- If doing a dark shadow, only satisfying assignments make sense.

Above summarised by theorems

\[(\exists x. L(x) \land U(x)) \equiv \text{“real shadow” (if exact)}\]
\[(\exists x. L(x) \land U(x)) \implies \text{“real shadow”}\]
\[\text{“dark shadow”} \implies (\exists x. L(x) \land U(x))\]
For solving “typical”, interactive goals, the Achilles Heel of this algorithm is the requirement to convert to DNF. This happens in two places.
The Trials of DNF

For solving “typical”, interactive goals, the Achilles Heel of this algorithm is the requirement to convert to DNF. This happens in two places.

Initially. Goals involving natural number subtraction can be particularly bad. Cooper’s algorithm (though generally slower) solves this faster than Omega:

!m n. 0 < m \(\land\) 0 < n \(\implies\) ((PRE m = PRE n) = (m = n))
The Trials of DNF

For solving “typical”, interactive goals, the Achilles Heel of this algorithm is the requirement to convert to DNF. This happens in two places.

- Initially. Goals involving natural number subtraction can be particularly bad. Cooper’s algorithm (though generally slower) solves this faster than Omega:

  \[ \neg m \land n. \ 0 < m \land 0 < n \implies ((\text{PRE } m = \text{PRE } n) = (m = n)) \]

- With alternating quantifiers. \( \forall x. \exists y. P(x, y) \) will be converted to \( \neg \exists x. \neg \exists y. P(x, y) \). When the innermost quantifier is eliminated, the negation must be pushed in over it, and everything converted back to DNF.
The importance of “gist”

Pugh and Wonnacott talk about using a gist operation to simplify terms of the form

\[ P \land (\exists x. \ldots) \]

where the \ldots gets rewritten while assuming \( P \).

This is contextual rewriting with the theorem

\[(P \Rightarrow (Q \equiv Q')) \Rightarrow (P \land Q \equiv P \land Q')\]

Implementing this well is vital to good performance. It made a difference of an order of magnitude on a VCG-generated goal of Peter Homeier’s.
Summary

- The Omega Test is an extension of the well-known Fourier-Motzkin variable elimination technique.

- In all those cases where the existing, incomplete Fourier-Motzkin variable elimination (FMVE) for integers works, Omega Test will work just as well.

- Omega is also complete, so it can additionally solve goals with any arrangement of quantifiers.

- Implemented in the Kananaskis release of HOL.