On the Implementation of Cylindrical Algebraic Coverings for Satisfiability Modulo Theories Solving

Gereon Kremer, Erika Ábrahám, Matthew England and James H. Davenport

Cylindrical Algebraic Coverings in a nutshell

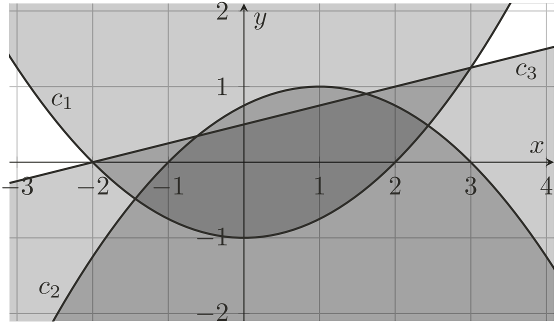
- Fix a variable ordering
- For the kth variable
  - Use constraints to exclude unsatisfiable intervals
  - Guess a value for the kth variable
  - Recurse to $k+1$st variable and obtain
    - a full variable assignment ($\rightarrow$ return SAT)
    - or a covering for the $k+1$st variable
  - Use CAD machinery to infer an interval from this covering
- Until the collected intervals form a covering for the kth variable

Called for the first variable, we get either

- a model, or
- a conflict (formulated as a covering).

An example

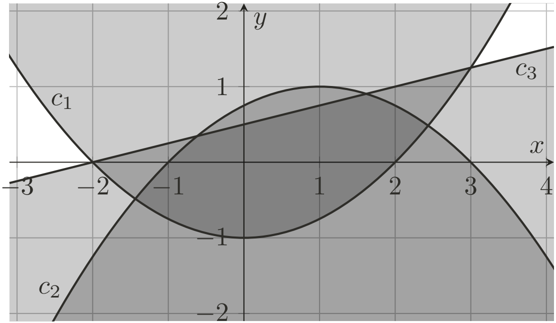
$$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$$



An example

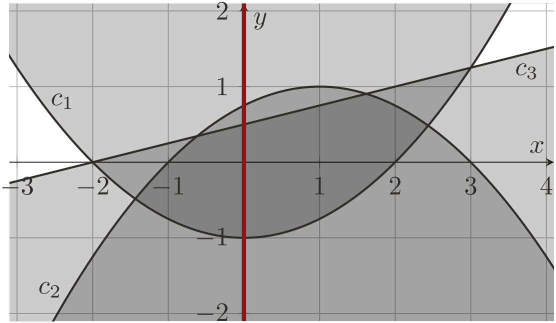
$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$

No constraint for x



An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$

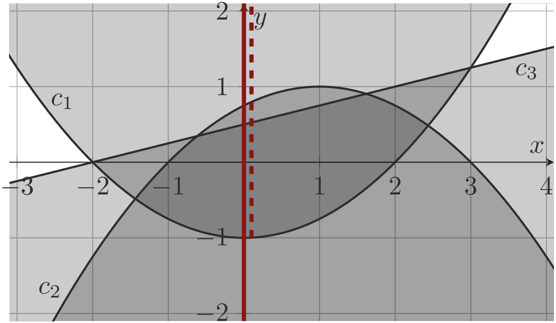


No constraint for x

Guess $x \mapsto 0$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



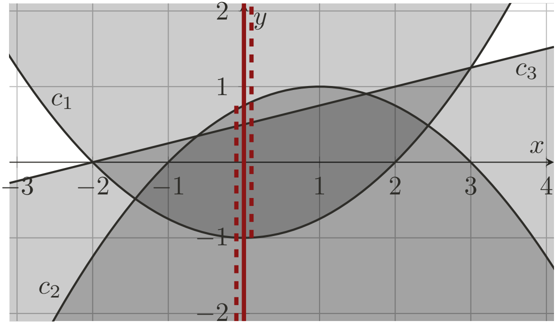
No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$$



No constraint for x

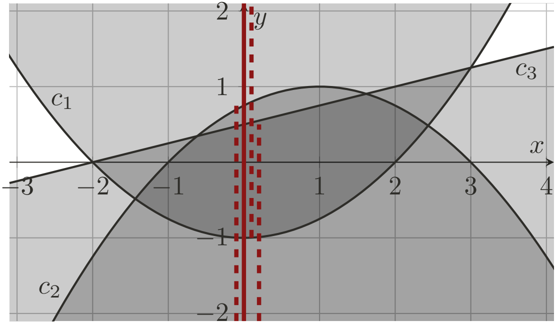
Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

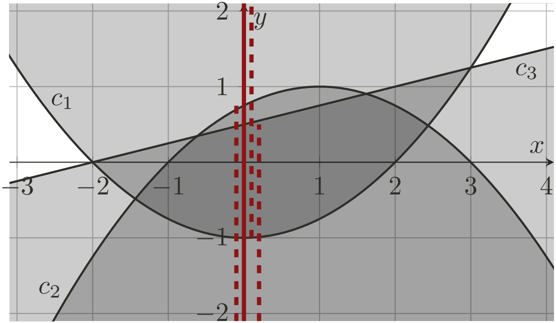
$c_1 \rightarrow y \notin (-1, \infty)$

$c_2 \rightarrow y \notin (-\infty, 0.75)$

$c_3 \rightarrow y \notin (-\infty, 0.5)$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

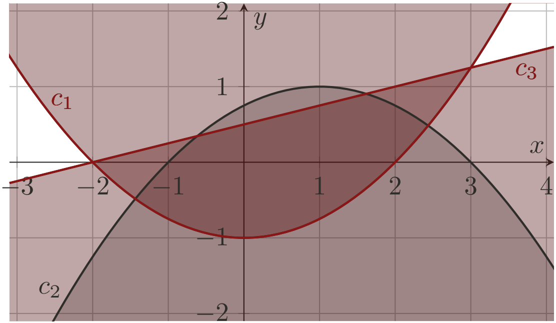
$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

$(-\infty, 0.5), (-1, \infty)$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

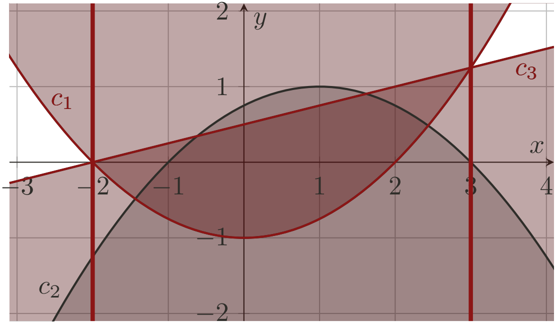
$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

$(-\infty, 0.5), (-1, \infty)$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

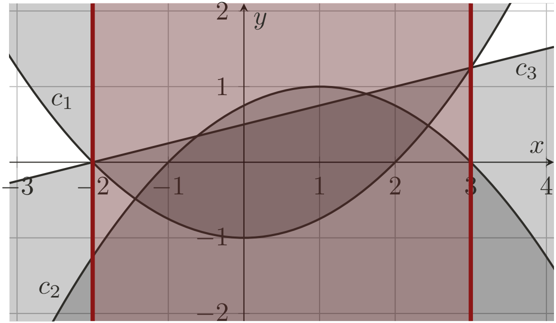
$(-\infty, 0.5), (-1, \infty)$

Construct interval for x

$x \notin (-2, 3)$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x
Guess $x \mapsto 0$
$c_1 \to y \notin (-1, \infty)$
$c_2 \to y \notin (-\infty, 0.75)$
$c_3 \to y \notin (-\infty, 0.5)$
Construct covering
$(-\infty, 0.5), (-1, \infty)$
Construct interval for x
$x \notin (-2, 3)$
New guess for x

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))
```

$\mathbb{I}$:= `get_unsat_intervals(s)`
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := `sample_outside`($\mathbb{I}$)
  if $i = n$ **then return** $(\mathrm{SAT}, (s_1, \dots, s_{i-1}, s_i))$
  (f, O) := `get_unsat_cover`($(s_1, \dots, s_{i-1}, s_i)$)
  if $f = \mathrm{SAT}$ **then return** (SAT, O)
  else if $f = \mathrm{UNSAT}$ **then**
    R := `construct_characterization`($(s_1, \dots, s_{i-1}, s_i), O$)
    J := `interval_from_characterization`($(s_1, \dots, s_{i-1}), s_i, R$)
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\mathrm{UNSAT}, \mathbb{I})$

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
    s_i := sample_outside(𝕀)
    if i = n then return (SAT, (s_1, ..., s_{i-1}, s_i))
    (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i))
    if f = SAT then return (SAT, O)
    else if f = UNSAT then
        R := construct_characterization((s_1, ..., s_{i-1}, s_i), O)
        J := interval_from_characterization((s_1, ..., s_{i-1}), s_i, R)
        𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

Real root isolation over a partial sample point

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
    s_i := sample_outside(𝕀)
    if i = n then return (SAT, (s_1, ..., s_{i-1}, s_i))
    (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i))
    if f = SAT then return (SAT, O)
    else if f = UNSAT then
        R := construct_characterization((s_1, ..., s_{i-1}, s_i), O)
        J := interval_from_characterization((s_1, ..., s_{i-1}), s_i, R)
        𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

The main algorithm

function get_unsat_cover($(s_1, \ldots, s_{i-1})$)

$\mathbb{I}$:= get_unsat_intervals(s)

while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**

  s_i := sample_outside($\mathbb{I}$)

  if $i = n$ **then** **return** $(\mathrm{SAT}, (s_1, \ldots, s_{i-1}, s_i))$

  (f, O) := get_unsat_cover($(s_1, \ldots, s_{i-1}, s_i)$)

  if $f = \mathrm{SAT}$ **then** **return** (SAT, O)

  else if $f = \mathrm{UNSAT}$ **then**

    R := construct_characterization($(s_1, \ldots, s_{i-1}, s_i), O$)

    J := interval_from_characterization($(s_1, \ldots, s_{i-1}), s_i, R$)

    $\mathbb{I} := \mathbb{I} \cup \{J\}$

return $(\mathrm{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

The main algorithm

`function get_unsat_cover`$((s_1, \ldots, s_{i-1}))$

$\mathbb{I} :=$ `get_unsat_intervals(s)`
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  $s_i :=$ `sample_outside`$(\mathbb{I})$
  if $i = n$ **then return** $(\mathrm{SAT}, (s_1, \ldots, s_{i-1}, s_i))$
  $(f, O) :=$ `get_unsat_cover`$((s_1, \ldots, s_{i-1}, s_i))$
  if $f = \mathrm{SAT}$ **then return** (SAT, O)
  else if $f = \mathrm{UNSAT}$ **then**
    $R :=$ `construct_characterization`$((s_1, \ldots, s$
    $J :=$ `interval_from_characterization`$((s_1,$
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\mathrm{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

The main algorithm

function get_unsat_cover($(s_1, \ldots, s_{i-1})$)

$\mathbb{I}$:= get_unsat_intervals(s)
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := sample_outside($\mathbb{I}$)
  if $i = n$ **then return** $(\mathrm{SAT}, (s_1, \ldots, s_{i-1}, s_i))$
  (f, O) := get_unsat_cover($(s_1, \ldots, s_{i-1}, s_i)$)
  if $f = \mathrm{SAT}$ **then return** (SAT, O)
  else if $f = \mathrm{UNSAT}$ **then**
    R := construct_characterization($(s_1, \ldots, s$
    J := interval_from_characterization($(s_1,$
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\mathrm{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

Extract interval from polynomials

The main algorithm

function `get_unsat_cover`$((s_1, \ldots, s_{i-1}))$

$\mathbb{I}$:= `get_unsat_intervals(s)`
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := `sample_outside`$(\mathbb{I})$
  if $i = n$ **then return** $(\text{SAT}, (s_1, \ldots, s_{i-1}, s_i))$
  (f, O) := `get_unsat_cover`$((s_1, \ldots, s_{i-1}, s_i))$
  if $f = \text{SAT}$ **then return** (SAT, O)
  else if $f = \text{UNSAT}$ **then**
    R := `construct_characterization`$((s_1, \ldots, s$
    J := `interval_from_characterization`$((s_1,$
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\text{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

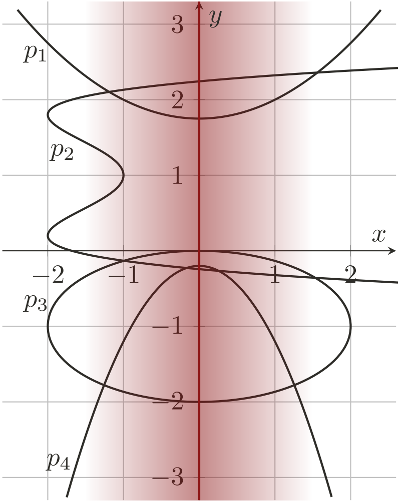
Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

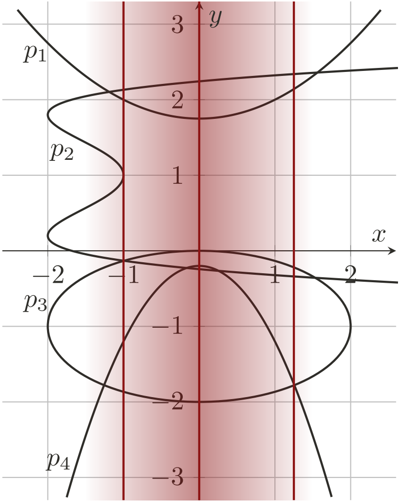
Extract interval from polynomials

construct_characterization



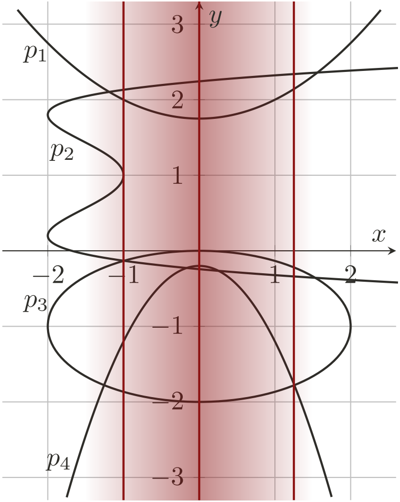
Identify region around sample

construct_characterization



Identify region around sample

construct_characterization

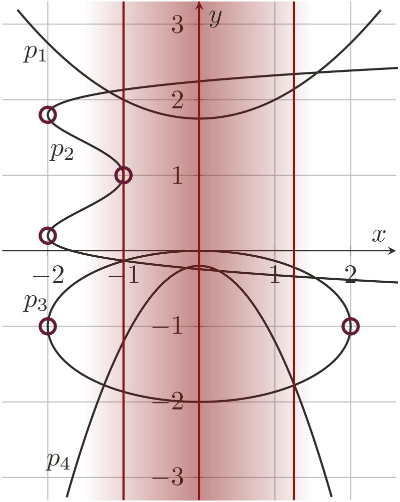


Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

construct_characterization

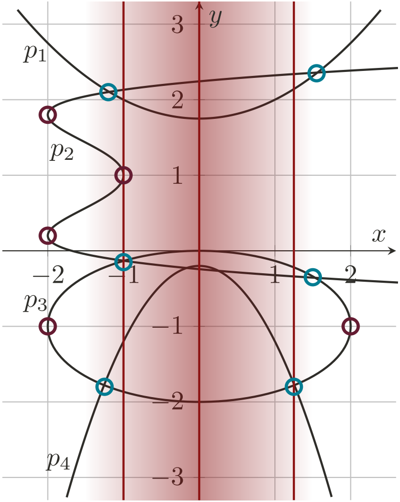


Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

construct_characterization

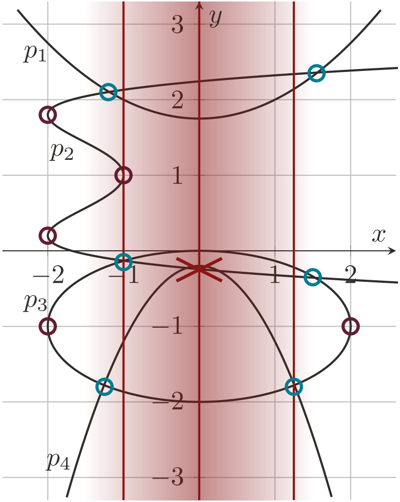


Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

construct_characterization



Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

Improvement over CAD:

- Resultants between neighbouring intervals only!

On the implementation in cvc5

- Heavily based on LibPoly [Jovanovic et al. 2017]
- Implements Lazard's lifting [Lazard 1994] using CoCoALib [Abbott et al. 2018]
- Different variable orderings based on [England et al. 2014]
  Nothing spectacular, though
- Generates infeasible subsets
- Allows for partial checks
  Not useful in our context as lemmas are nonlinear
- Supports mixed-integer problems
- Experimental support for incremental checks
  No performance benefit observed

Courtesy of cvc5

- Integrated with linear solver [Cimatti et al. 2018]
  Incremental linearization scheme
- Generates formal proofs
  Not detailed enough yet for automated verification
- Arbitrary theory combination
  It just works!
- Applicable to quantified problems
  No changes necessary!

Experiments

First implemented in SMT-RAT

- Preliminary implementation (no incrementality, no optimizations)
- Easily outperforms regular CAD (from [Kremer et al. 2020])

Second implementation in cvc5: winner of QF_NRA @ SMT-COMP 2021

Solver	SAT	UNSAT	overall	
cvc5	5021	5377	10398	90.0%
yices (NLSAT)	4904	5437	10341	89.5%
z3 (NLSAT)	5093	5195	10288	89.1%
SMT-RAT	4438	4435	8873	76.8%
cvc5 (without CAC)	3283	5385	8668	75.0%
cvc5 (no nl reasoning)	2203	3271	5474	47.4%

References I

- John Abbott, Anna M. Bigatti, and Elisa Palezzato. "New in CoCoA-5.2.4 and CoCoALib-0.99600 for SC-Square". In: SC^2. FLoC. Vol. 2189. July 2018, pp. 88–94. URL: http://ceur-ws.org/Vol-2189/paper4.pdf.
- Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. "Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions". In: ACM Transactions on Computational Logic 19 (3 2018), 19:1–19:52. DOI: 10.1145/3230639.
- Matthew England, Russell Bradford, James H. Davenport, and David Wilson. "Choosing a Variable Ordering for Truth-Table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition". In: ICMS. Vol. 8592. 2014. DOI: 10.1007/978-3-662-44199-2_68.
- Dejan Jovanovic and Bruno Dutertre. "LibPoly: A Library for Reasoning about Polynomials". In: SMT. CAV. Vol. 1889. 2017. URL: http://ceur-ws.org/Vol-1889/paper3.pdf.
- Gereon Kremer and Erika Ábrahám. "Fully Incremental Cylindrical Algebraic Decomposition". In: Journal of Symbolic Computation 100 (2020), pp. 11–37. DOI: 10.1016/j.jsc.2019.07.018.
- Daniel Lazard. "An Improved Projection for Cylindrical Algebraic Decomposition". In: Algebraic Geometry and its Applications. 1994. Chap. 29, pp. 467–476. DOI: 10.1007/978-1-4612-2628-4_29.